SCHOOL BOT • RISCBOT • LAPTOP BOT • MOVIE BOTS

SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

March 2006

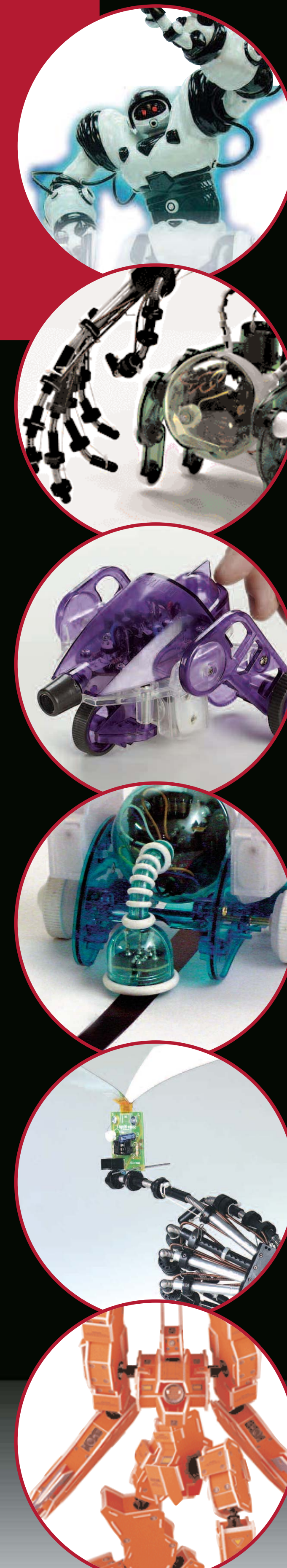
Assimilate Your Robot With the BoRG BOARD

Jasper the 3D Movie Bot

Give Your Microcontroller More Drive

Portrait of the Artist as a Young Robot

www.servomagazine.com



SERVO

Features & Projects

ON THE COVER

Meet the BoRG — The Boise Robotics Group

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. APPLICATION TO MAIL AT PERIODICALS POSTAGE RATE IS PENDING AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

03.2006
VOL. 4 NO. 3

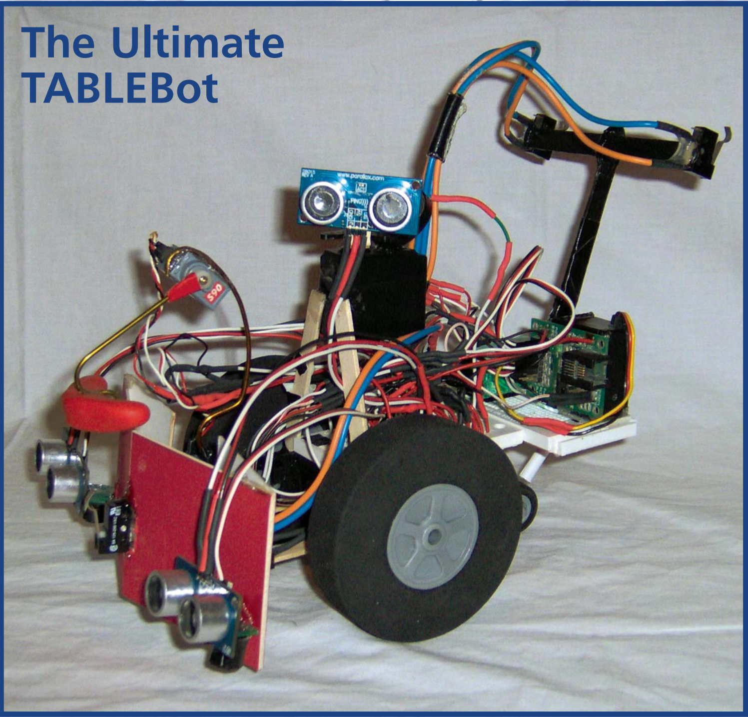
Departments

Columns

Coming 04.2006

Robots Who Live With People

The Ultimate TABLEBot



Mind / Iron

by R. Steven Rainwater

How many times have members of your local robot group debated long into the night about machine consciousness, emotional robots, or some other hot-button issue, only to conclude that no one agreed on the definition of a word? Recently, I ran across an online debate — impressive in both size and scope — that had me laugh out loud. The debate centered on whether or not the T-800 Terminator robot of movie fame was a cyborg. The controversial point was whether the definition of cyborg included machines with human parts, as well as humans with machine parts. While reading the debate, it became clear to me that neither side was aware of the origin of the word cyborg, or the long history of its changing meanings.

Norbert Weiner coined the term cybernetics in the summer of 1947. What did he mean by it? He says, "We have decided to call the entire field of control and communication theory, whether in the machine or in the animal, by the name of *Cybernetics*, which we form from the Greek kybernetes or steersman."

The interesting point is that just as any machine with self-regulatory feedback mechanisms is a cybernetic machine, so humans are cybernetic organisms because our bodies and brains include self-regulatory feedback mechanisms. Cybernetics is the study of these mechanisms. This means you and I, just as we were born, without the need for machine parts of any kind, are cybernetic organisms.

Two things affected the meaning of cybernetics after 1948. The first was that, while the books and papers published in the field analyzed both animals and machines, the general public was more fascinated by descriptions of "self-replicating machines" and "learning machines." They tended to ignore the biological aspect of cybernetics. Second, the use of the word among scientists declined, as it was replaced by terms like complexity theory or dynamic systems theory which meant much the same thing. By the 1970s and 1980s, scientists used the term less frequently, and the general public increasingly misused it to describe intelligent or life-like machines.

To complicate things, a new word came out of NASA in 1960. Dr. Manfred Clynes and Nathan S. Kline combined the words *cybernetic organism* to form cyborg. They also gave this word a completely new meaning. Dr. Clynes' webpage summarizes: "His concept of a cyborg was of a symbiosis between a person and a machine, creating an interaction that would enhance life, such as a man and his bicycle, but in other pursuits, such as space travel."

So, according to Clynes, humans were no longer to be considered cybernetic organisms unless they existed in symbiosis with a machine of some kind. Symbiosis did not mean man and machine were one, they merely worked together, the way we work with a cell phone or a laptop.

Not surprisingly, it didn't take long for cyborgs to jump from NASA to science fiction. Robert Heinlein narrowly beat Frank Herbert to the

Mind/Iron Continued →

Published Monthly By
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Product Order Line **1-800-783-4624**
www.servomagazine.com

Subscriptions
Inside US **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux
Michele Durant

first recorded use in print. Heinlein uses the term cyborg several times in the serialized version of *The Moon is a Harsh Mistress*, which appeared in *If* magazine beginning in December 1965 (the term itself didn't occur until the March 1966 installment of the story). Heinlein used the word in a negative sense far more extreme than anything Clynes imagined: "Man, I saw one disturbing report. It was claimed that attempts are being made at the University of Peiping to combine computers with human brains to achieve massive capacity. A computing Cyborg."The usage in science fiction varied widely but most often referred to a fusion of man and machine, instead of the symbiosis Clynes proposed.

How did the word move from science fiction to mainstream usage? In 1972, Martin Caidin published a science fiction novel titled, simply, *Cyborg*. The book became the basis of a television series about an injured test pilot named Steve Austin, who is fitted with machine parts to become a superhuman government agent. The series was, of course, *The Six Million Dollar Man*. Cyborg, along with bionic, became household words. Cyborg came to mean not Weiner's idea of a cybernetic organism, nor Clyne's idea of a human existing in symbiosis with a machine, but what the TV told us it meant: a biological human improved through the integration of machine components.

The moral of this story for the robot philosophers of local robot groups is to make sure you agree on the definition of your words before getting too deep into debate over how they apply to the robots (or cyborgs) you're building. **SV**

Attention roboteers!

We want to hear from you! Do you have a great bot that you would like to share with the world? Send us a couple of pictures of your latest project, and we'll be happy to show it off in our "Menagerie" department. Don't forget to include a few words about how you built it and what went into it. Email them to **menagerie@servomagazine.com**

BIO-FEEDBACK

Dear SERVO:

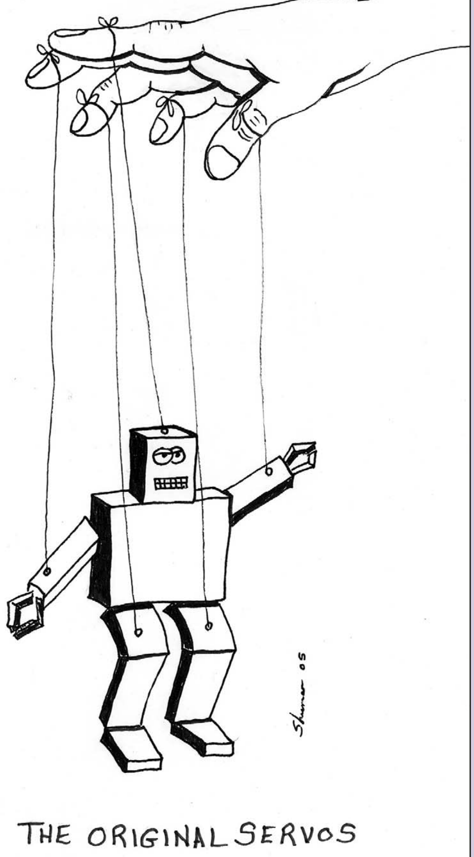
I have just finished reading the latest *SERVO* issue and there are several errors with the RS V2 review that I find extremely troubling. The writer claims that Robosapien is RSV1 and this is wrong. It is RS. Worse than this tech boo-boo is the problem with the entire text — it reads like PR BS.

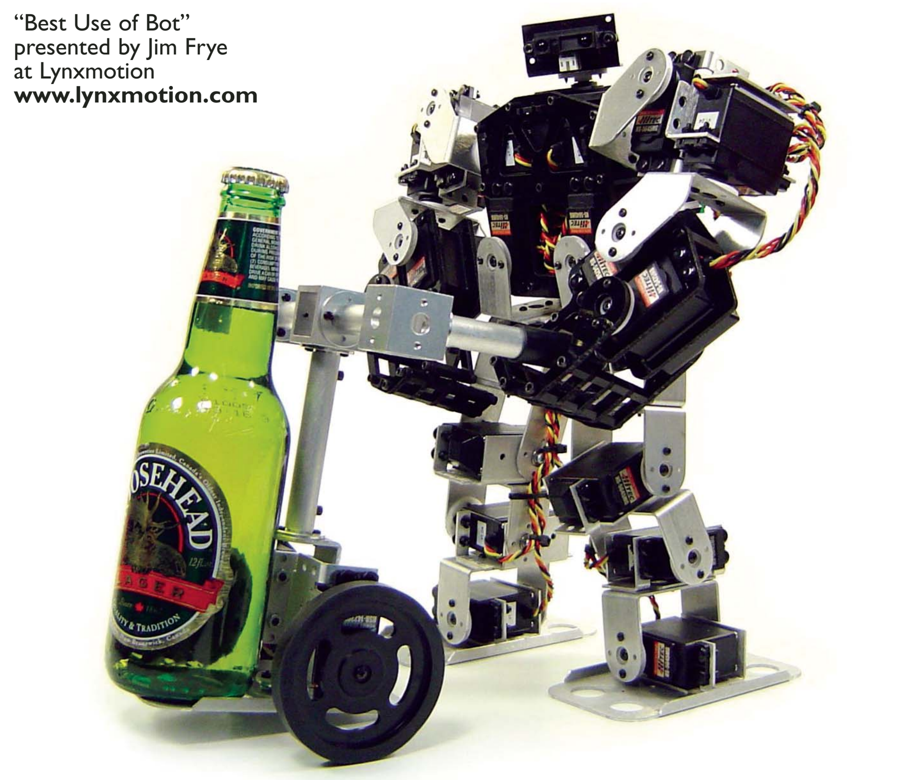
There are MANY problems with the RS V2, the biggest of which is the repeated failure of the dual hip motor gearboxes. This is such a problem that sales are lacking AND most robotic workers are shunning the design.

Unfortunately, these design/production problems are being seen throughout the entire new WowWee product line and, as such, NONE of the current models has enjoyed the same success as RS. In my opinion, the emperor has no clothes and, as such, I wouldn't want this type of "glee club" review reflecting poorly on *SERVO*.

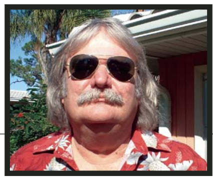
Anonymous
via Internet

by J. Shuman





"Best Use of Bot"
presented by Jim Frye
at Lynxmotion
www.lynxmotion.com



Robytes

by Jeff Eckert

Are you an avid Internet surfer who came across something cool that we all need to see? Are you on an interesting R&D group and want to share what you're developing? Then send me an email! To submit related press releases and news items, please visit www.jkeckert.com

— Jeff Eckert

Improved Speech Technology

Sensory, Inc.'s RSC-4x chip provides enhanced speech recognition. Photo courtesy of Sensory.

In January, Sensory, Inc. (**www.sensoryinc.com**), announced some enhancements to its RSC-4x product, which it bills as the world's best selling speech recognition chip. Using the company's FluentChip™ technology, it is capable of recognizing dozens of speaker-independent words or phrases in a single set and can also recognize speaker-dependent recognition words (customized by the user) or perform speaker verification for biometric security.

The chips not only "talk and hear," they have an embedded microcontroller that can act as the brain for consumer electronic products, and they can record speech and play back MIDI or digital music. New capabilities include beat detection (picking up the amplitude of different sounds and reacting to them with a movement or display function), beat prediction (the chip comprehends a recurring beat to know how react to it, as for dancing), sound sourcing (using a second microphone to allow the processor to locate the sound of a human voice), talk-back (replying with human speech), pitch detection (voice analysis to figure out what pitches are being sung), and sing back (combining talkback and pitch detection allows a robotic creature or avatar to imitate a person singing).

RSC chips, which are reported to sell for only about $2 in manufacturing quantities, are used in products by Hasbro, JVC, Kenwood, Mattel, Sony, Toshiba, and others, so you can expect the improved technology to appear in consumer products soon.

"Mighty Mouse" Survives Work in Deadly Radiation

Bob Anderson demonstrates capabilities of the "Mighty Mouse" robot. Photo by Randy Montoya, courtesy of Sandia National Laboratories.

One of the things they do at the Department of Defense's White Sands Missile Range lab (**www.wsmr.army.mil**) in New Mexico is to irradiate circuit boards and vehicles to see if the electronics can stand up to the kind of radiation that would be present if someone set off a nuclear weapon nearby.

This involves a cylindrical cobalt-60 radiation source that's approximately the size of a salt shaker. However, no one really wants to pick one of them up — given that they give off enough gamma rays to kill you in about half a minute — so the sources are moved around pneumatically in metal sleeves, sort-of like how the bank does transactions at the drive-up islands. Usually, about 20 psi of pressure will do the trick, but the story has now emerged about how one of the cobalt cylinders got stuck after ramming into a signal switch, and even 1,000 psi wouldn't get it loose. The result was 21 days of blaring alarms and flashing warning lights until a robot, affectionately dubbed M2, for Mighty Mouse, was sent in by Sandia National Laboratories (**www.sandia.gov**) to save the day.

M2 — a 600-lb, five-foot robot — rolls on treads, can maneuver around obstacles, and has a multijointed gripper arm that is suitable for operating drills and screwdrivers to dislodge the cylinder. The Sandia team estimated that the robot could survive only about 50 minutes in the radiation environment before its own circuits started to go bad, so the plan was to have it move quickly to drill a hole and remove the switch so the cobalt cylinder could be removed.

Unfortunately, an hour and a half later, several attempts at dislodging the switch had failed, M2's lower drive portion was no longer working, and he had to be dragged out by a rope. The story gets complicated, with many trips to Lowe's and Home Depot for improvised repair parts, but the bottom line is that M2 succeeded, after three days, in unscrewing six bolts that held a steel plate over the switch, removing it, and thereby solving the problem. The warning lights and horns — which could be heard for miles around — finally went off. The facility design is being evaluated to see how similar incidents can be prevented.

Dual Theorems Produce Better Bots?

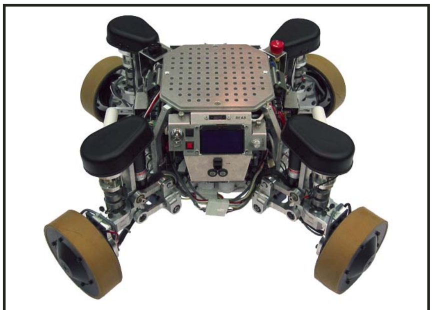
This multiple-platform robot design hints at a new class of robots that maintain their strength, even when damaged. Photo courtesy of Purdue University.

The folks at Purdue University (**www.purdue.edu**) have recently published information about a new approach to designing better structures, machines, and robots. The concept combines mathematical theorems use by civil engineers (statics) and mechanical engineers (kinematics). The theorems offer promise in creating a new class of "multiple-platform robots" that maintain their strength even when damaged or otherwise compromised.

According to Purdue Associate Professor Gordon R. Pennock, "These new theorems represent a common language and provide an understanding of what we call the duality between kinematics and statics. The practical result is that engineers can use this knowledge to design better structures and better machines."

In the example shown, we have a 12-legged robot that has two flat platforms: a lower platform that has six legs standing on the ground and an upper platform that is connected to the ground by four legs and to the lower platform by two legs. Apparently, the advantage is that, even if this type of bot is damaged or restricted in its motion capabilities, it will nevertheless remain stable and functional. Perhaps a less theoretical model will make the advantages more obvious. This one just looks like two coffee tables mating.

Omnidirectional Robot Available



The Azimut 2 — from RoboMotio — is an omnidirectional platform that can carry up to 34 kg (~91 lb). Photo courtesy of Robomotio.

An interesting development from RoboMotio (**www.robomotio.com**) and the Research Laboratory on Mobile Robotics and Intelligent Systems (affiliated with the University of Sherbrooke, in Quebec), is the Azimut 2, which moves on four independently directed wheels, each with its own brushless motor and gearbox. The overall size is 60 by 52 by 29 cm (approx. 24 by 21 by 12 in), it weighs in at 35 kg (approx. 94 lb), and can carry almost its own weight.

According to the company, Azimut 2 can change the direction of its wheels by more than 180 degrees, so it can move sideways without changing its heading. This makes is particularly agile in restricted environments.

It is powered by two 24V battery packs or an external power tether and controlled by an embedded Pentium M-based Mini-ITX computer. No price was given for the machine, but Robomotio operates mostly on research and military contracts, so one probably won't be appearing in your living room anytime soon. **SV**

Programmable Logic

by Gerard Fonte

The goal of this bimonthly column is to provide a basic understanding of the various programmable logic techniques.

There are a lot of powerful low-cost components available today that are rarely considered by hobbyists — and even some engineers — because of unfamiliarity.

You have to be comfortable with the idea and concepts of programmable logic before you will be likely to employ them.

In this part, we are going to introduce the concepts necessary for understanding PAL (Programmable Array Logic) architecture and memory logic which is the basis for most ASIC (Application Specific Integrated Circuit) approaches. We will also look at memory arithmetic. Two basic circuits will be examined to see the strong points and weak points of each approach.

Sample Circuits

To start out, let's look at Table 1. This is the standard decoding pattern for a common seven-segment LED display. The decimal value is defined in the left column and the appropriate segments are lit if there is a "1" in the box. If there is a "0" in the box, the segment is off. A binary value that corresponds to the digital value is provided for convenience. At the bottom of the table is a reference diagram for identifying the physical layout of the segments. Lastly, there is a "SAMPLE" row with "CASE 1" and "CASE 2" for segments "C" and "E," respectively. These are the two basic decoding circuits that we will be designing. None of the other segments will be examined.

The seven-segment decoder circuits were chosen because they have no inherent pattern to them. This means that we will be finding general solutions to arbitrary logic. The very good thing about general solutions is that they can be applied to every

```
      A
  ********
  *      *
F *      * B
  *  G   *
  ********
  *      *
E *      * C
  *      *
  ********
      D
```

SEGMENT IDENTIFICATION

TABLE 1. This shows the logical function of a seven-segment decoder. We will example only segment C and segment E. The physical layout is shown in the diagram at the bottom.

Value	Binary Code	Seg A	Seg B	Seg C	Seg D	Seg E	Seg F	Seg G
0	0 0 0 0	1	1	1	1	1	1	0
1	0 0 0 1	0	1	1	0	0	0	0
2	0 0 1 0	1	1	0	1	1	0	1
3	0 0 1 1	1	1	1	1	0	0	1
4	0 1 0 0	0	1	1	0	0	1	1
5	0 1 0 1	1	0	1	1	0	1	1
6	0 1 1 0	0	0	1	1	1	1	1
7	0 1 1 1	1	1	1	0	0	0	0
8	1 0 0 0	1	1	1	1	1	1	1
9	1 0 0 1	1	1	1	0	0	1	1
Sample				Case 1		Case 2		

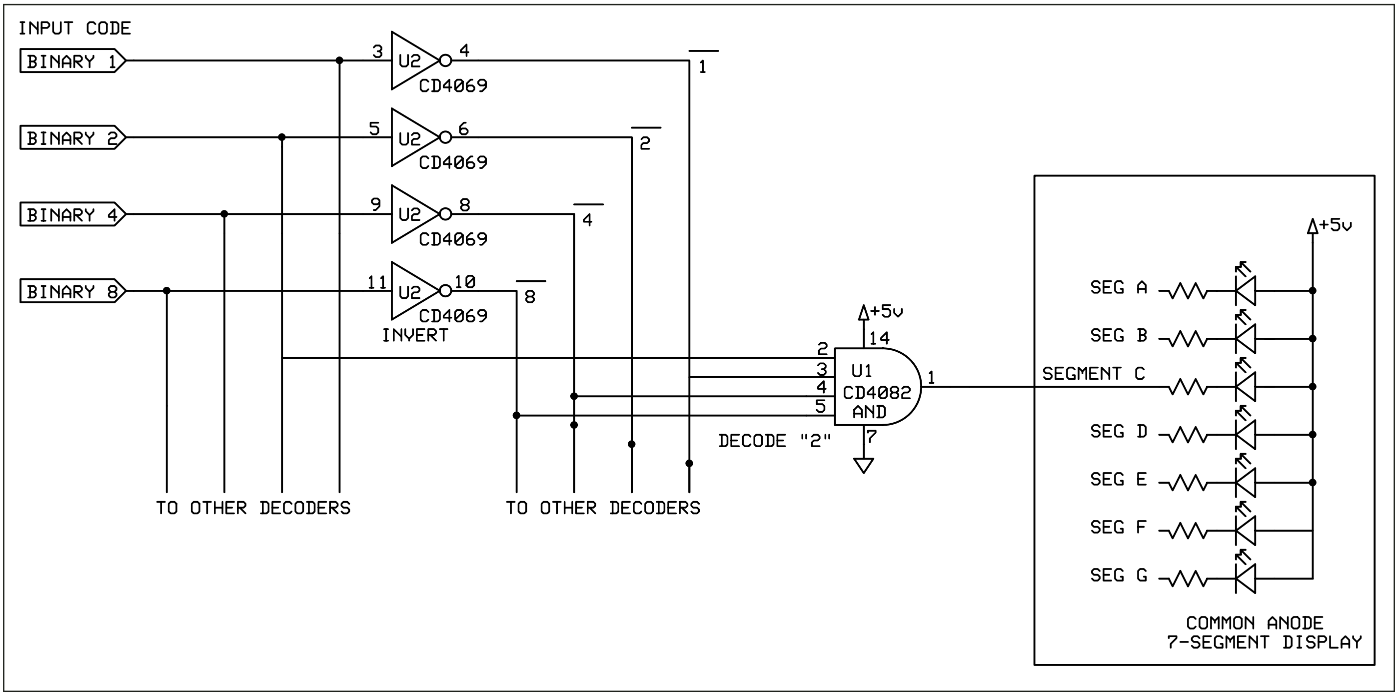


FIGURE 1. The decoding logic for segment C is shown here. All the inputs are inverted here in order to save inverters needed for other decoders.

design application. Sometimes there are special cases where the general solution is not as fast or economical (in terms of the number of logic gates) as other approaches. However, these special cases are limited in their usefulness. We will see that the general solutions have few limitations and can be extremely flexible and powerful.

Figure 1 is the decoding logic necessary for decoding the value "2." Segment "C" of a seven-segment display is always on except for the value "2." Therefore, the only value that must be decoded is "2." Since we are using common anode displays, a high output (or +5 volts) will turn off the LED. (A low output will allow current to flow and light the LED.) If you examine the Figure 1 schematic, you will find that the only time the inputs to the AND gate are all high is when the binary value input is "0010" (a value of "2"). Note that four inverters are used. This approach actually saves gates because we will have to build nine more decoders for all the other decimal values that are possible. If separate inverters were used for every decoder, we would have to use dozens of them. This is more apparent in Figure 2.

Figure 2 is the decoding logic necessary for decoding all the values needed to control segment "E" in the seven-segment display. As you can see, it's much more complicated. Five binary values (or states) must be decoded: "1," "3," "4," "5," and "8." Each value is decoded with a four-input AND gate in a manner very similar to Figure 1. Since we want the LED to light whenever any of these states is present, we OR these signals together. Because there are no five-input OR gates commonly available, we are forced to use two four-input gates, as shown.

The inverter at the output of the OR-gate that drives the LED is needed because we are using a common anode display which lights when the LED terminal is pulled low. (*Note that 13 inverted signals are used for this segment alone. So, inverting the binary values right at the beginning really does save lots of gates.*)

PAL Logic

This general procedure of inverting all the inputs, ANDing the proper signals, and then ORing them is used over and over in digital design whenever multiple outputs from the same inputs are needed. This is the basic design approach for all PAL devices.

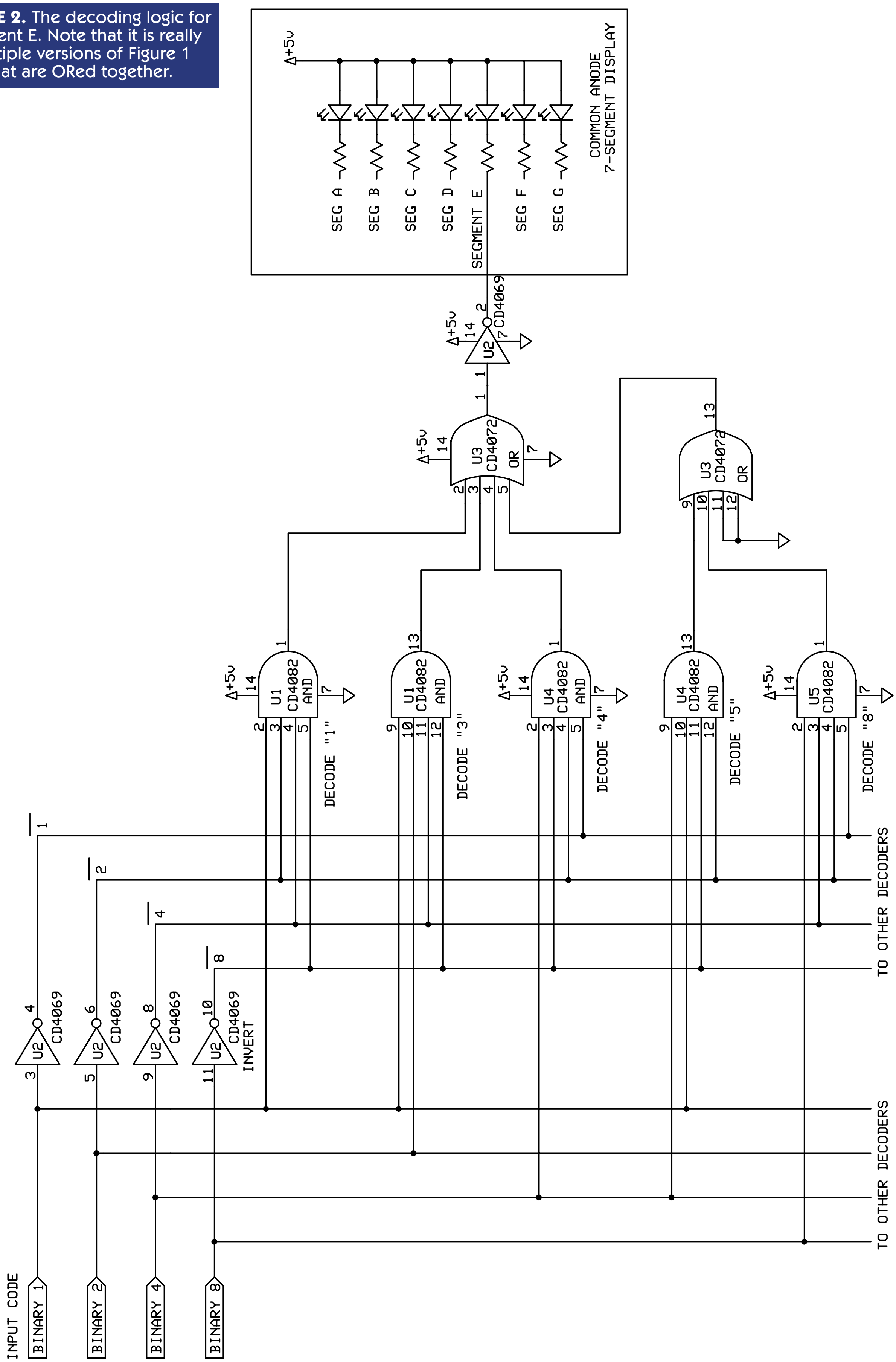
Note that there a number of varieties of "PAL" families. Besides PALs — which are one-time programmable — there are GALs (Generic Array Logic) and PEELs (Programmable Electrically Eraseable Logic). Both of these can be erased and reprogrammed many times. There are other PLDs (Programmable Logic Devices) available that can be very simple or highly complex. Next time, we will examine these different families in more detail.

It's important to note that the INVERT/AND/OR approach always works for decoding signals. But consider the situation where you want to decode values "1," "3," "5," "7," and "9." You can use the same circuit in Figure 2 and just change the AND inputs. That will certainly work. Alternatively, you can use the actual binary LSB (Least Significant Bit). It provides exactly the decoding pattern we want (refer to Table 1). Obviously, using the LSB directly is much more efficient than using a large number of gates to re-create the same signal. This means that we still have to think about what we are doing.

Memory Logic

Figure 3 illustrates how PROMs (Programmable Read Only Memory) can be used to implement logic func-

FIGURE 2. The decoding logic for segment E. Note that it is really multiple versions of Figure 1 that are ORed together.



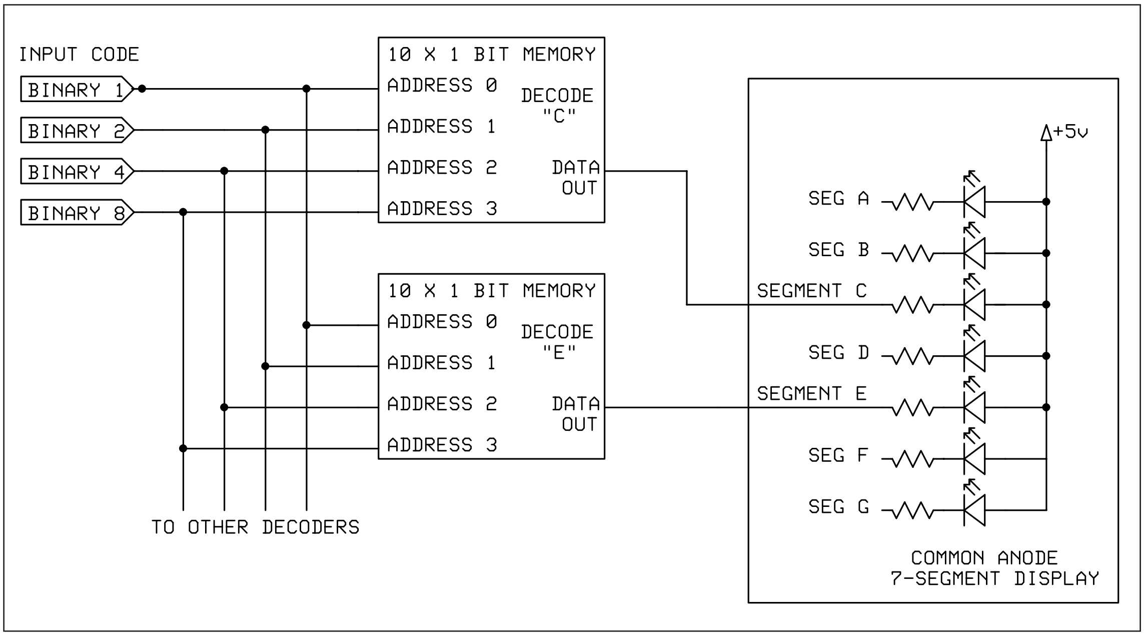


FIGURE 3. The hardware for a memory logic approach is the same for either segment C or E (or any other segment). However, the memory must be programmed properly.

tions. At first, it seems somewhat odd to think of memory as logic. It's certainly not intuitive. But if you stop and think about it, it does work. Instead of going through all the exercises of decoding the inputs and using inverters, AND gates, and OR gates, you simply build a look-up table.

The input value can certainly be considered as an address or pointer to an entry in a table. And the table result can be 0 or 1, as desired. So all the decoding logic goes away. Instead, we are left with a "black box" where we apply an input code and get the desired results. Of course, for this to work properly, the PROM must be programmed with the proper values.

Figure 3 also shows that the physical decoding circuit for segment C is exactly the same as the circuit for decoding segment E. This is an extremely useful property for two main reasons. The first is that any changes in logic don't change the hardware. You only have to change the PROM. And if the PROM is erasable, all you have to do is reprogram it with the new values.

There is no cutting and jumpering of traces and no need to re-layout the printed circuit board (PCB). Additionally, this approach provides a fixed-size solution to any logic pattern. This is useful if you are laying out a PC board or a silicon wafer.

The second important property is that the decoding speed is identical for any and all logic functions. This virtually eliminates glitches and spikes that can wreak havoc with downstream logic. Compare Figure 3 with Figures 1 and 2. Figure 1 has two basic gate delays. Figure 2 has five gate delays (four if an OR gate with more inputs can be found). This difference in delays is not important with just LEDs, because human vision is relatively slow. But if additional logic downstream was used, steps would have to be taken to be sure that the timing difference wouldn't cause problems.

Obviously, there is the difficulty of finding a 10-location by one-bit memory device. None exist. However, we can improve things somewhat by combining all the decoders and using only one memory because each decoder uses the same four input values. This means

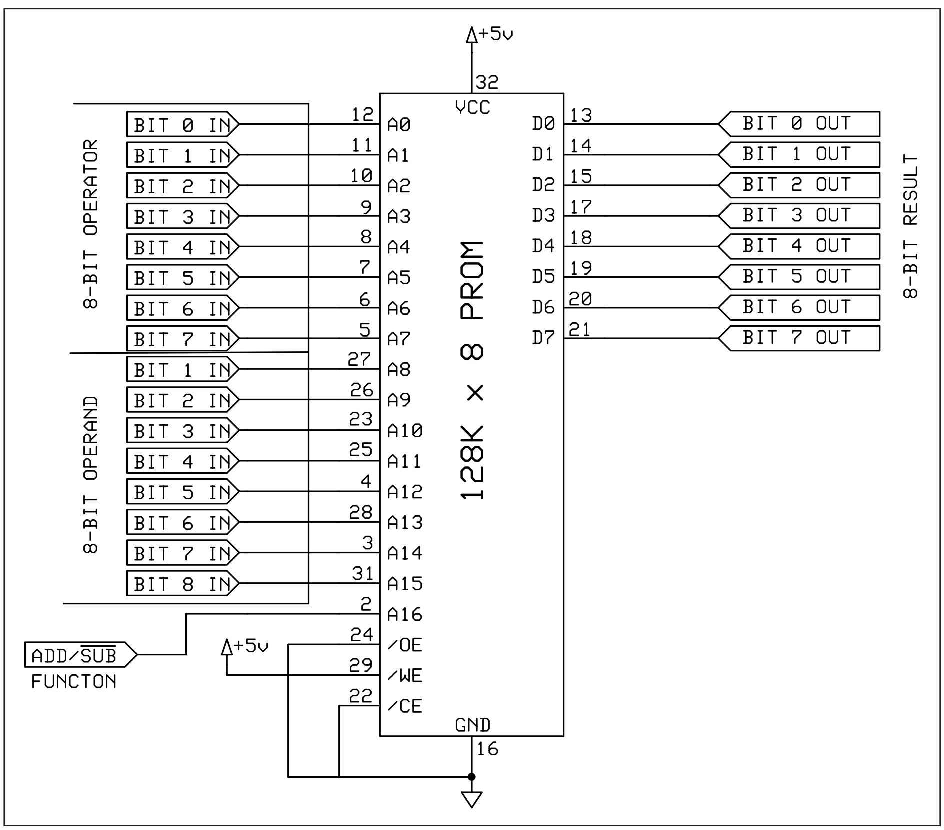
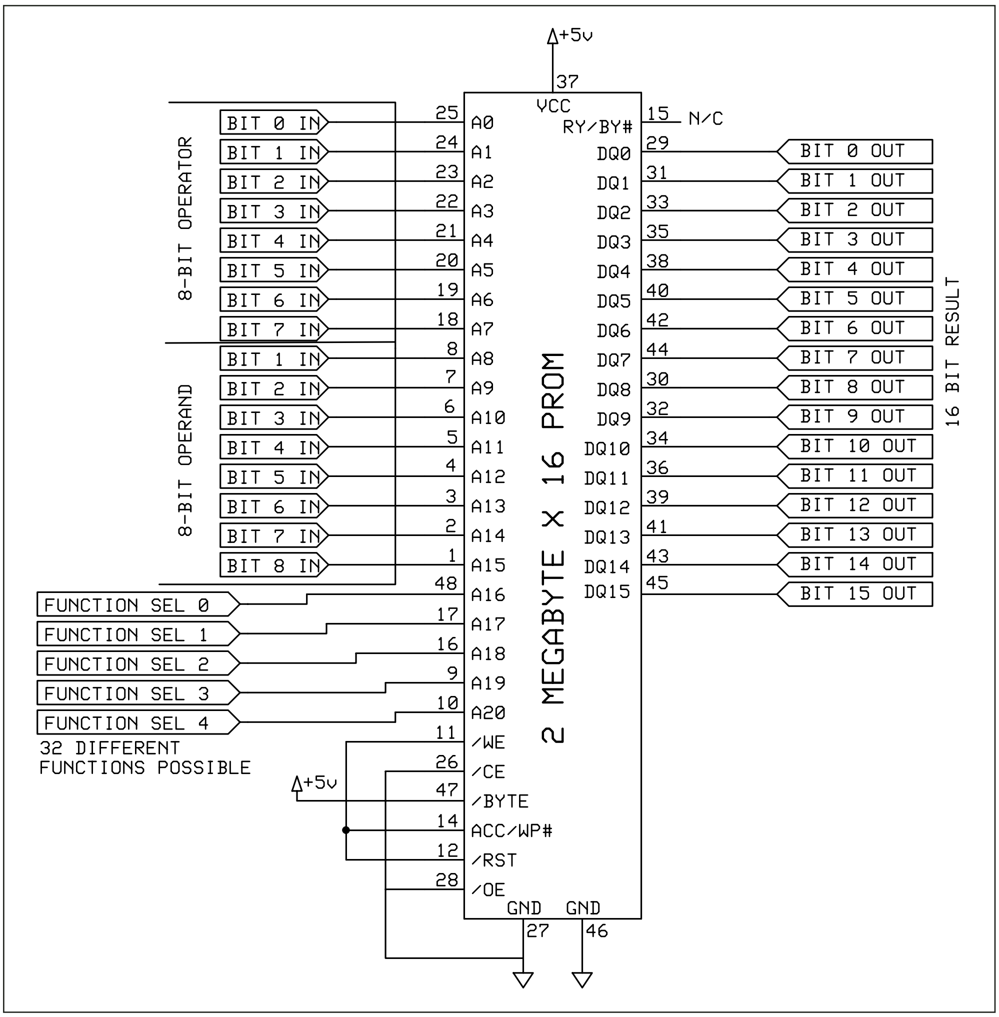


FIGURE 4. A simple eight-bit ADD/SUBTRACT memory arithmetic circuit can be implemented with a single chip. It has limited usefulness.

that we can use a 10 by seven-bit memory. That's a slight improvement. There are lots of eight-bit memories available, but they are much larger than only 10 locations. That's the main problem with PROM logic. There is so much memory wasted when conventional devices are used.

However, most ASICs use a memory-logic approach. They can have hundreds or thousands of these small memories that are used as logic blocks on one IC. The ASIC is programmed by loading these memories with the data that matches the desired logic function. These on-board memories can be very fast. Some ASICs have memory speeds of just a nanosecond or two. We will discuss ASICs in much more detail in a future installment.

FIGURE 5. A simple and very powerful 16-bit memory arithmetic calculation circuit also uses just a single inexpensive chip. The cost is about $3.50 and it produces results in 90 nS.



PROM Arithmetic

While PROM logic often makes inefficient use of the memory, PROM arithmetic can be very efficient. Look at Figure 4. This is an add/subtract function with one-byte inputs and a one-byte result. The whole function is contained on one standard 128K by eight PROM. Again, the PROM must be programmed with the proper values. But figuring that out is really pretty easy.

Note that a one-byte operator and operand require two bytes of addressing, or 64K addresses. Since the add function requires 64K, we can use the other half of the 128K PROM for the subtract function. But adding and subtracting bytes can be done so much easier with a simple microprocessor (µP), why consider this approach?

Figure 5 shows why this approach can be extremely useful. Here we have a standard two-megabyte by 16 PROM. (For example, we could use the Atmel AV49BV4096-90 Flash memory IC which costs about $3.50.) We can create 32 different mathematical functions with 16-bit accuracy. Choose any function you like, or even design your own functions, like (A+B)/(A*B). If you have a single operand function — like cosine — you can use a full 16-bit input value. And here's the really nice part, every function will take exactly the same time to execute.

With the Amtel part, this speed is 90 nS. Since it's so much faster than most of the common µP, there's no waiting. Output your values and function choice from your µP, then read back the result as fast as you can. If you need to do significant geometry to figure out where your robot is, or you need fast results to keep your walker balanced, the PROM arithmetic method can be an ideal solution.

Computer Logic

It must be noted that a computer can also be used as logic. In this case, you use software to examine the computer's inputs and create an appropriate output. However, it is very slow and limited. It is mentioned here only for completeness, because it has little utility in real life.

Conclusion

PALs use a general invert/AND/OR approach that works quite well for small-scale, specialized logic needs. Next time, we will take a closer look at PALs and related devices. Memory logic is used in ASICs which generally have much greater capabilities than PALs. These will also be discussed in a future article. PROM arithmetic can be extremely fast and cost-effective. But you have to program it properly. Having a basic understanding in the operational principles of these programmable logic techniques will allow you to be more comfortable in employing them. **SV**

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

Jasper — the 3D Movie Bot

It Films Carnivorous Plants!

Using simplicity to solve an extremely complex problem and create something timeless to boot.

The Carnivore and a Robotic John Ott — a Marriage Made of Mindstorms: The Journey Begins!

It all starts with something quite removed from robotics: carnivorous plants; so far removed, in fact, that these pernicious plants don't even eat robots — just bugs and such.

Eventual roboticist by way of need, Mike Wilder has been growing carnivorous plants for more than a decade. His "garden" is the birthplace of hundreds of meat eaters of known types, and hybrids he creates himself. Mike writes and lectures about his passion, as well.

Because Mr. Wilder's fascination has presented a great tool for teaching kids about the importance of plant life, the idea blossomed for a subsequent teaching platform — a time-lapsed movie in 3D that climaxes in a carnivorous plant eating and growing.

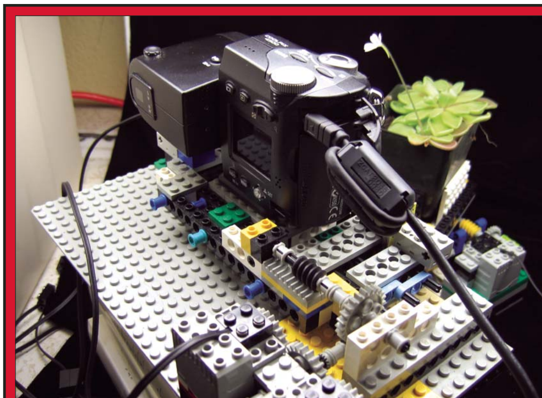
Having made 2D time-lapsed films of his plants with his still camera as of '03, Mike figured it was time to move to 3D.

"I thought, man, it would be so amazing if I could make 3D films of them growing while they were rotating. I am a rather serious student of time-lapse film, and time lapse with rotation is very rare. In macro 3D, I believe it is unprecedented," says Wilder.

Necessity is the Mother of Robots. It's True.

In the summer of '04, film and robot hobbyist Mike Wilder set out to produce a 3D movie about carnivorous plants. Remember the ads in the comic books for seeds to grow the Venus Flytrap? Wait until you see the bug-extinguishing flora in Mike's film clip (see Resources at the end of this article), which is only a portion of the whole DVD presentation.

Note the twin motors driving the juxtaposition module. The hope was that this would reduce fatigue on the motors, and save the time lapse in case one motor failed during a shot (redundancy).

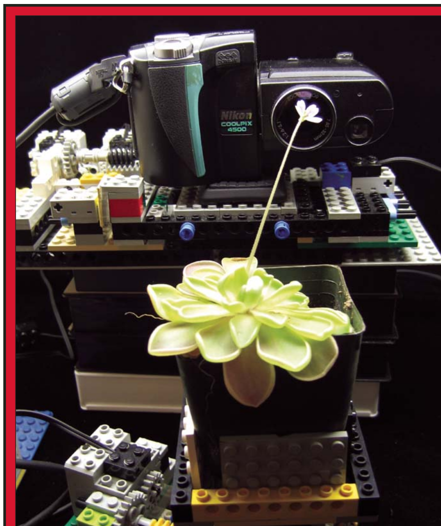


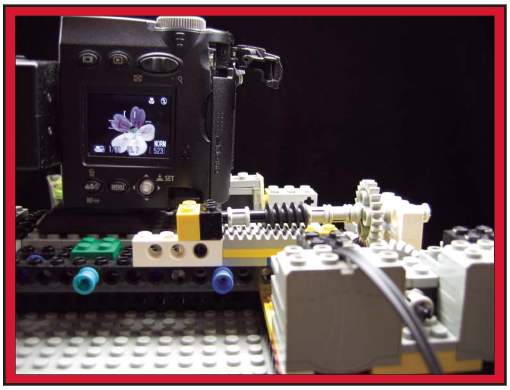
The Problem: Conventional filming in 3D calls for two cameras taking left and right view shots of the person or, in this case, plant being filmed. This sounds good so far, but there is a downside to this method, one that would be a particular obstruction for Wilder's work.

The two-camera filming technique makes it optically impossible to get very close up, i.e., macro view 3D shots.

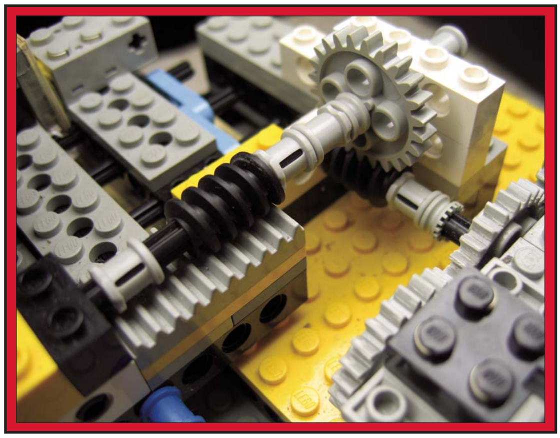
The Solution: Wilder's Jasper robot (his first robot ever; how's that for hitting one out of the park?). The Mindstorms-based bot moves a single camera to get the views usually

Here, we are facing the robot. The camera is mounted in the juxtaposition module, and the rotation module (in the foreground) holds a carnivorous plant.





The camera is aimed at the tiny flower. Note the rack at the end of the gear train, which causes the camera to slide from side to side in that plane.



Close-up of the gear train on juxtaposition module. Worm to 24 tooth to worm to rack.

achieved by two cameras.

The robot operates the camera shutter release and rotates a Lego turntable that houses the carnivorous plant it is filming. By rotating the turntable in modest increments of 1/10th of a degree, the robot was able to create a time-lapsed close-up in 3D of the plant rotating as it grows.

The Saga of Jasper Unfolds

Every great achievement is fraught with some difficulty or discouragement. When Wilder first compiled the idea for his Juxtapositioning Automatic Stereo Pair Emulation Robot (JASPER), he shared his forthcoming invention with an expert in both photography and physics.

This expert — also a friend of Wilder's — conveyed that it would be an extremely trying exercise to try to get any machine to operate the way Wilder had in mind with the precision needed to pull it off. In particular, Wilder's friend believed this would be especially challenging with a Lego-based robot because of the likely gear lashing that would occur.

The Problem in Detail

We perceive three-dimensional objects by combining the views we get separately from our right and left eyes. 3D movies are based on this understanding. They require two separate pieces of film, one with a left eye view and one with a right eye view.

This requires two cameras, sitting about 2.5 inches apart, the average distance that our pupils are apart from each other. The problem is that you can't take close-ups of something small and have both cameras be able to get the shot while sitting 2.5 inches apart.

The Solution in Detail

Wilder built a Lego Mindstorms

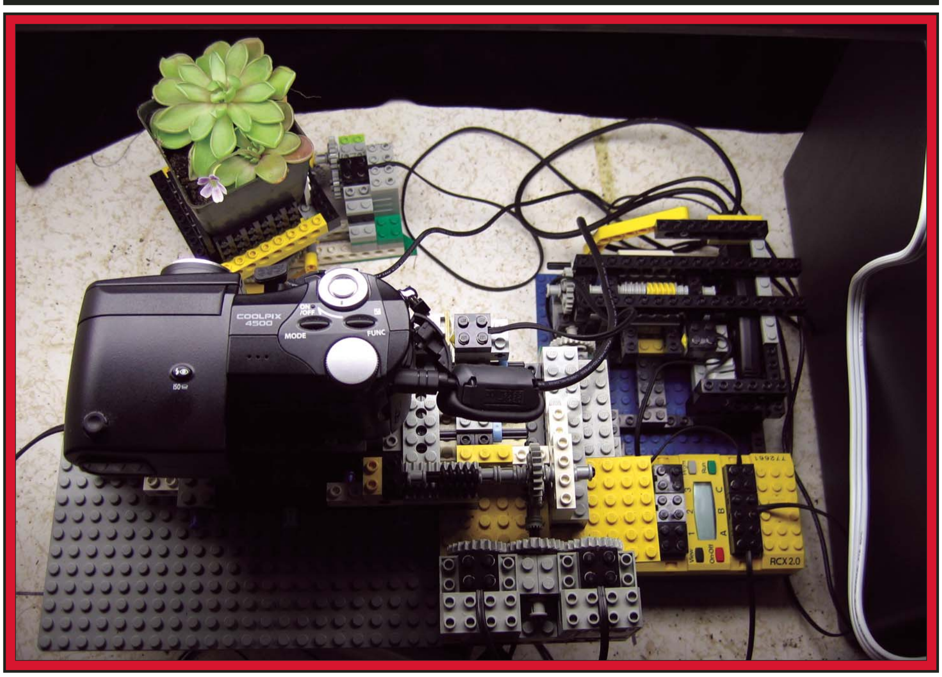
ROLE 'EM

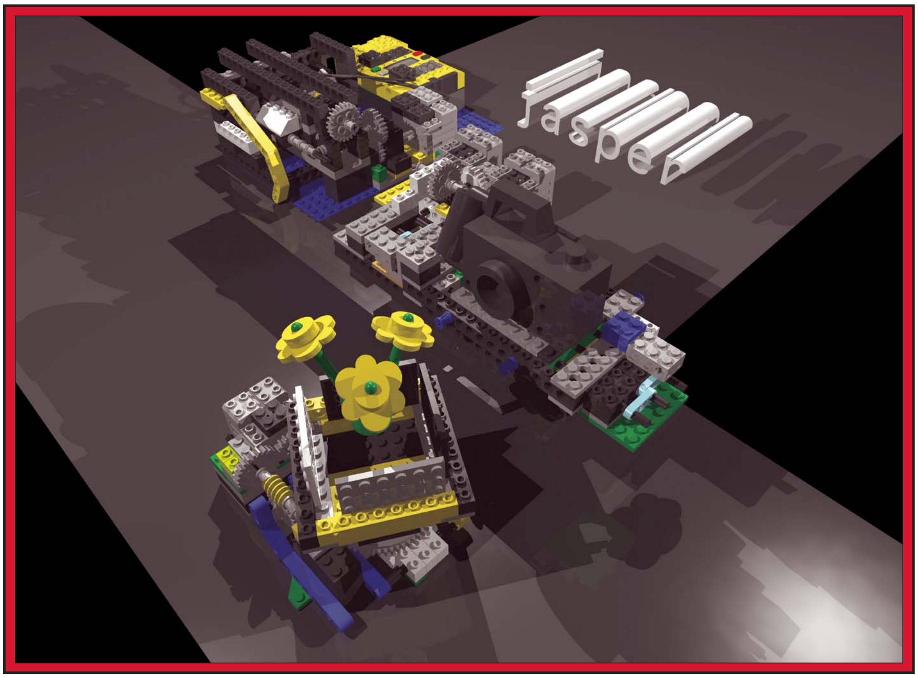
Rather than producing actual movies, Jasper takes stills with a common digital camera — the Nikon Coolpix 4500. Jasper produces about 2,000 stills in a week's time — a thousand from each angle view.

Roboticist Mike Wilder takes these images from the camera's memory card and processes them into 1,000 left and right angle pairs of the same point in time for the carnivorous flower. These images then become 1,000 3D images, which produce 33 seconds of 3D video.

"I used the freeware 'Stereo Photo Maker' to convert each left-right pair into one 3D image, and then I used Adobe Premiere to convert the 1,000 3D images into ca. 33 seconds of 3D video (each image is one frame in a 30 frames-per-second movie)," says Wilder.

This is a view of the whole system in action. On the right is the cable release module with RCX, on the left is the juxtaposition module with camera, and on the top left is the rotation module with plant.





A computer rendering of Jasper, with the three modules.

robot that could move his Nikon coolpix 4500 digital still camera back and forth 6-mm in a single plane to get the two views while also being close up.

Jasper consists of three modules, measuring 22 x 5 inches if you were to put them all in a row, with a maximum height of around 6.5 inches. "The weight is two to three pounds. That's just a guess — the camera weighs a pound, and there aren't all that many Legos. The microcontroller is light since it has no batteries," says Wilder.

The Jasper robot solution required a solution of its own to a new problem it created. With the likely gear lashing, the movements of the robot could err by as much as .5-mm from the two points it would need to stop at, 6-mm apart from each other. This would make for a blurry movie.

Mike's answer was another simple solution. He put touch sensors at the left and right positions where he needed the robot to stop in its movements back and forth across that 6-mm plane. "When the camera was in the right spot, the sensor would close, and the motor would stop," says Wilder.

From there, the robot would take the needed picture from that position, the motor would reverse and stop when the other sensor closed. The picture would be snapped from that angle and the process would repeat.

PARTS-N-PROGRAMMING

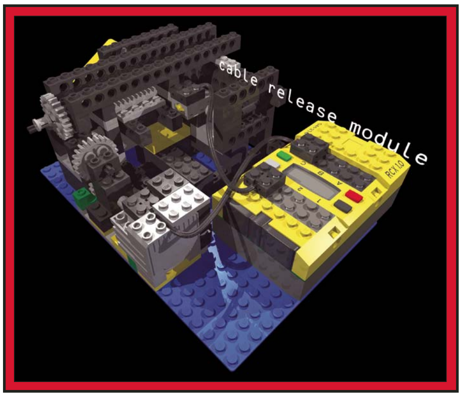
For this project, an RCX 1.0 was needed for its DC adapter port. "I acquired one of these from eBay (a battery-powered RCX would not have run for weeks at a time), says Mike Wilder, photographer and roboticist.

Mike used parts from two Mindstorms kits along with extra worm gears and gear plates from Bricklink. The rest were Lego parts — 396 of them — except for the camera and remote camera cable. The Jasper build called for four Mindstorms motors among its three modules.

"Jasper was programmed with the stock Lego programming language," says Wilder. The programming addressed how long to film the subject and the interval between left and right pairs of images.

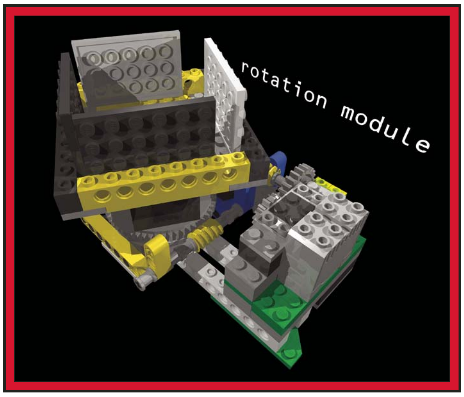
"Once Jasper has the right program for the task, I arrange the plant how I want it in front of the camera, and then hit 'run.' Come back a week or two later, remove the camera's memory card, and Jasper's work is done," concludes Wilder.

Here, you can see the output of Jasper. The Venus Flytrap's trap is about one inch long, so we really are shooting close up! So, what Jasper actually produces is shown in Figure A and Figure B. The distance moved was 6 mm. Obviously, the two pictures are very similar. But, you need these two views to make one 3D picture, which is shown in Figure C. That one is made from the right-left pair with software, totally separate from the robot.



This is the cable release module.

This is the juxtaposition module.



This is the rotation module.

This stopping solution worked, but there were a few other problems: taking the picture itself and rotating the plant. Mike equipped the robot with a remote cable (Nikon MC-EU1) to remotely push the shutter release button on the cable, so as not to shake the camera by pushing the camera-based shutter button directly.

The button on the cable is a two-stage button. It must be pressed only slightly at first to wake the camera, then fully to take the picture. The robot's finger presses that button. Wilder used a very high-torque gear train so the finger moved very, very slowly.

The finger length was specified so that the initial pressing of the button, only part way, happened exactly when a touch sensor was closed. The finger later pressed the button fully to take the picture.

RESOURCES

Web home of Jasper.
www.3dsyndrome.com

The entire time-lapsed film of a carnivorous plant taken by Jasper, a Lego Mindstorms robot, using a single still camera.
www.3dsyndrome.com/moviepage.html

Order film on DVD. Comes with two pairs of 3D glasses.
www.3dsyndrome.com/purchase.html

How to build Jasper yourself.
www.3dsyndrome.com/automata.html

Lego Mindstorms
http://mindstorms.lego.com/?domainredir=www.legomindstorms.com

NW Film Festival
www.nwfilm.org/

3D Center for Art and Photography
www.3dcenter.us

Bricklink
www.bricklink.com/

The Turning Point

The final apparatus was the turntable to turn the plant one revolution per month, discontinuously, according to Wilder. The turntable had to stop for the camera to take the left and right angle view images, wait 10 minutes, and start again. The intervals between shots were 10 minutes.

Without using sensors, the Lego Mindstorms turntable was geared to move in precise 1/10th of a millimeter steps every 10 minutes.

Roboticist Mike Wilder, seated with the Jasper robot modules.

Jasper's Robot Qualifications

"Someone once asked me: 'What does the robot know? How is it different from a programmable coffee maker?' He was trying to argue, indirectly, that Jasper isn't a robot. My answer was that Jasper doesn't really know anything, except he knows when he has touched himself. The robot that built your car has only that kind of knowledge, too," says Wilder, explaining his response.

Jasper is simply aware that one of its touch sensors has closed. Jasper is not a robot based on its brilliance. However, Jasper has several things in common with much more complex robots.

Jasper is a programmable machine with sensors that can change behavior based on sensor inputs in order to achieve a goal.

While Jasper uses a simple set of responses to a very limited set of inputs, it efficiently performs its job, says Wilder. "It reliably collects photographic data 24 hours a day for weeks at a time. This is a task no

human could perform," he adds.

A Star is Bloomed

Of all the carnivorous plants Wilder could have chosen, he selected one with movements as rare as the robotic film process he had selected to capture them. The 3D film clip mentioned previously is of a 4-mm carnivorous plant opening — a plant that only opens for a few hours on a single afternoon once a year! The complete film is available on CD.

No, Seriously

Only now are viewers and critics convinced that the Jasper film project could be done. At midpoint in making his 3D masterwork, Wilder presented an initial piece of footage — a demo disk — to the 3D Center for Art and Photography. The footage included animations of Jasper in action.

Basically, they thought Mike was nuts, responding with something to the effect that you can't make a 3D movie by moving a still camera around with some toys. "People underestimate what you can do with Legos," says Wilder. "Even the 3D experts couldn't believe that a simple solution (just sliding a camera back and forth) was possible," he adds.

"Later, I gave a lecture to a general audience. Part of the lecture discussed the 3D movie I was working on, and how I was filming it with a Lego robot. Again, people thought I was joking!" says Wilder.

Conclusion

Historically, with this achievement Wilder has accomplished very precise data collection in a way that no one has achieved before by using something as simple as a Lego Mindstorms robot and a digital still camera.

The film has been screened at the NW Film Festival, as well as at the 3D Center for Art and Photography. Jasper itself has been demo'ed at PDXBOT 05, a Portland area robotics expo. SV

This month, we'll be hacking into the popular Scribbler robot from Parallax. This is our first time working with a robot from Parallax, and we must say that from the very start we were blown away by the quality and accessibility of the Scribbler Robot. It is a quality robot kit that comes at a particularly accessible price — $99 — so that makes it a member of the exclusive under $100 club, along with other great kits like the Robosapien.

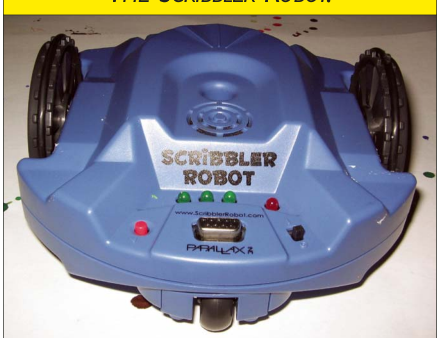
The greatness of the Scribbler only begins with the price tag, because this is a little robot with a big vision behind it. The package proudly proclaims that the Scribbler is appropriate for kids ages eight and up, and while that might seem a bit young for a budding roboticist, after working with the Scribbler for a little while we were confident that even preteen roboteers could learn a thing or two from this artistically inclined robot.

Primary Colors

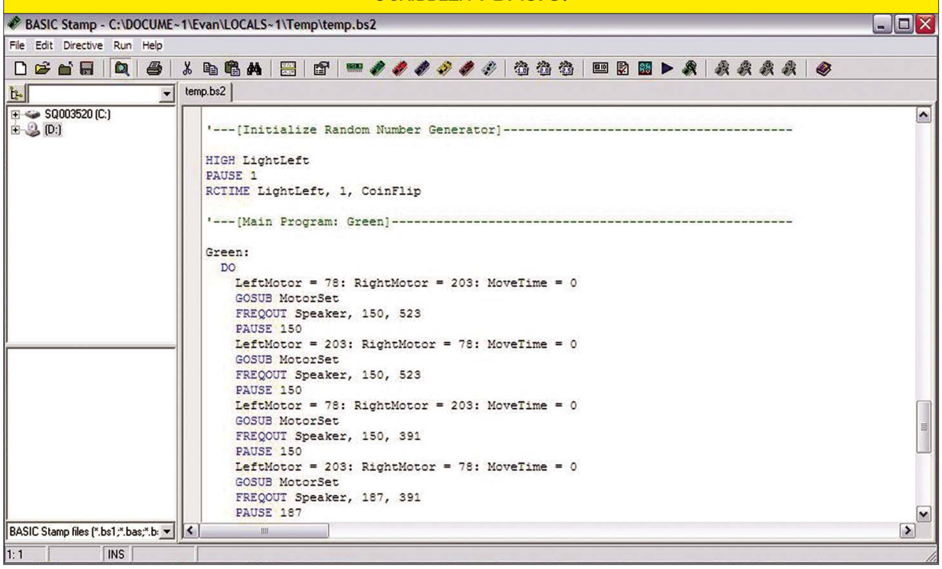
Even without any hacking or programming, the Scribbler can be one entertaining little robot. All you need are six AA batteries to get started — the Scribbler comes fully-assembled with eight demo modes. The demos are an excellent demonstration of the abilities of the versatile kit, encompassing everything from light-seeking and line-following to the very entertaining "ambulance mode," where the Scribbler flails around with sirens blaring while avoiding obstacles.

To make these various behaviors possible, the Scribbler comes with a

THE SCRIBBLER ROBOT.



SCRIBBLER PBASIC.


BASIC Stamp - C:\DOCUME~1\Evan\LOCALS~1\Temp\temp.bs2

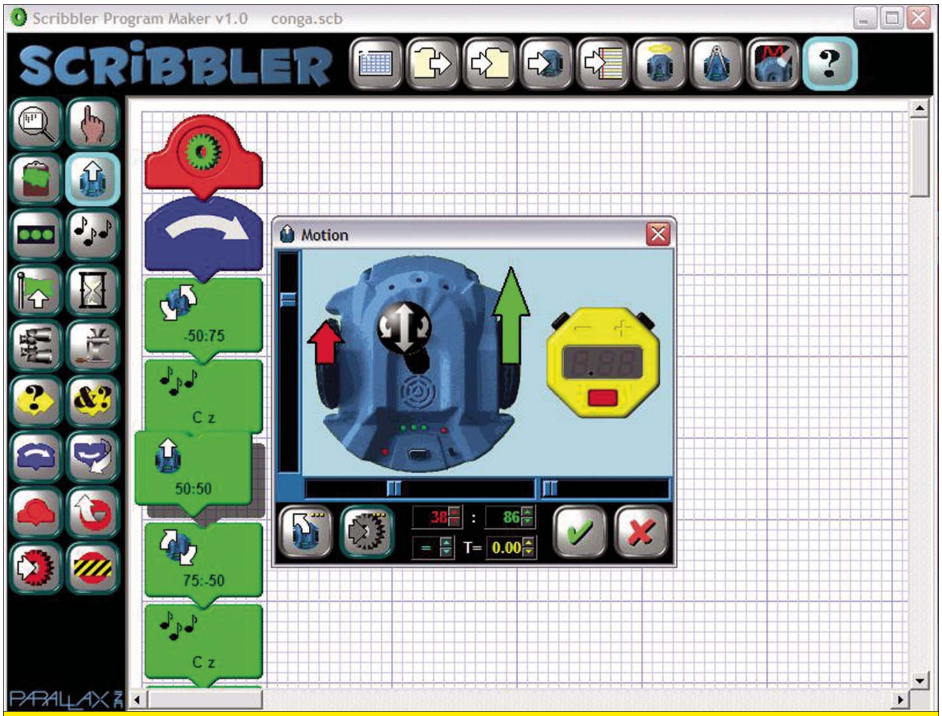
File Edit Directive Run Help

```
'---[Initialize Random Number Generator]---------------------------------------

HIGH LightLeft
PAUSE 1
RCTIME LightLeft, 1, CoinFlip

'---[Main Program: Green]-----------------------------------------------------

Green:
  DO
    LeftMotor = 78: RightMotor = 203: MoveTime = 0
    GOSUB MotorSet
    FREQOUT Speaker, 150, 523
    PAUSE 150
    LeftMotor = 203: RightMotor = 78: MoveTime = 0
    GOSUB MotorSet
    FREQOUT Speaker, 150, 523
    PAUSE 150
    LeftMotor = 78: RightMotor = 203: MoveTime = 0
    GOSUB MotorSet
    FREQOUT Speaker, 150, 391
    PAUSE 150
    LeftMotor = 203: RightMotor = 78: MoveTime = 0
    GOSUB MotorSet
    FREQOUT Speaker, 187, 391
    PAUSE 187
```



The Scribbler's GUI.

full suite of sensors. The Scribbler comes equipped with three photo resistor light sensors, infrared emitters and detectors, a stall sensor, and line-following sensors. The helpful *Scribbler Robot Start-up Guide* is choc full of colorful illustrations that explain how these sensors work in a way directed towards younger roboticists that still manages not to insult the intelligence of more experienced tinkerers.

Programming the Scribbler can be as easy or as complex as the user would like it to be. The Scribbler is programmed in PBASIC, and it comes with a text-based environment, as well as a graphical user interface to accommodate both programming wizards and novices. We've never used PBASIC before, so the GUI was a welcome way to introduce ourselves to the programming of the Scribbler robot.

The Scribbler's GUI is reminiscent of the LEGO Mindstorms programming environment, and it is similarly accessible. Users simply drag and drop commands, then click on the blocks to edit various parameters like motor speed and direction. The Scribbler GUI is slightly more complex than the LEGO programming by virtue of the ability to create and call various subroutines, which is an advantageous feature for the more experienced programmers using the kit.

With its range of programming abilities, the Scribbler kit creates an environment conducive to fostering programming growth and learning for its users, whether those users are beginners looking to scratch the surface of the programming world or veteran programmers looking for an entertaining platform where they can cut their coding teeth.

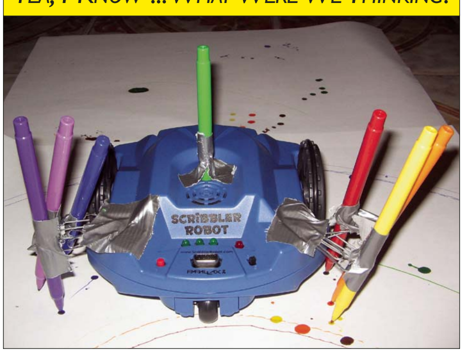
If It Isn't Baroque, Don't Fix It

Previous Scribbler projects featured in *SERVO* have involved some fancy programming, and we're happy to be left to a mainly mechanical hack. But what kind of hack would be useful for an artistically inclined automaton? A hack to diversify its repertoire, of course!

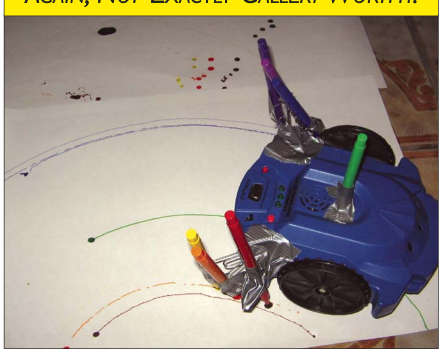
One way in which the Scribbler is limited in its artistic ability is through the less than intuitive way it grasps its medium of choice. The stock Scribbler holds its instrument with a simple hole that runs through the center of the Scribbler's body. The "pen holder" is really only suitable for the recommended implement — a Sharpie type felt tip pen. That limits the Scribbler to monochromatic drawings that bleed through thin papers, so there is definitely room for improvement.

The simplest and ugliest way in which we tried to address the problem was by using duct tape and paper clips. With a shameless lack of consideration for aesthetics, we duct taped paper clip tendrils to the sides of the Scribbler to allow it to wield up to seven colors (six colors in addition to the one that can be placed in the existing "pen holder"). What do you draw with seven colors? A rainbow, of course!

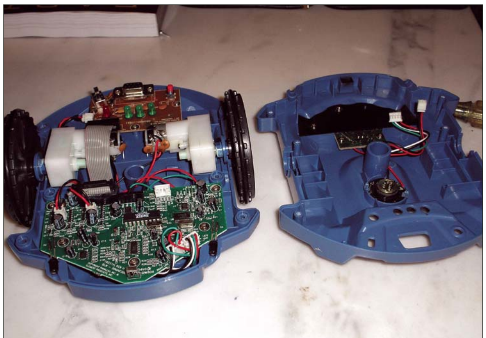
Yea, I Know ... What Were We Thinking?



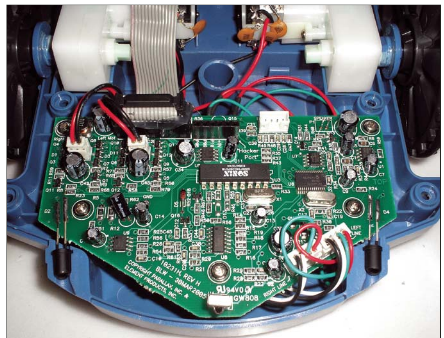
Again, Not Exactly Gallery Worthy.



The Hacker Port.



Ewww ... Scribbler Guts!



The Panel.

As you might expect, a rainbow drawn with markers held by paper clips and duct tape was not exactly gallery worthy. But that merely motivated us to find a better way.

A Discriminating Palette

Our paper clip and duct tape escapade might have seemed like a useless exercise, but it actually allowed us to pinpoint several weaknesses that we wanted to address with our hack. For one, the Scribbler was limited to utilizing a single implement at a time, and for two, the size and shape of that implement were dictated by the nonadjustable pen holder. Our silly rainbow shenanigans were actually an important step in the creative process — we were able to identify the criteria that went into what we judged as effective media handling (diversity and versatility), and this enabled us to give the Scribbler a discriminating mechanical palette.

To give the Scribbler more artistic finesse, we wanted our palette to be capable of holding several media of varying sizes while only pressing one to the paper at a time. An easy way to do this would be to craft a palette-like turntable with several "pen holders." To allow the multiple pen holders to hold media of varying sizes, the pen holders would have to be more than simple holes drilled out of our palette.

To find a way to make versatile pen holders, we looked to fast food soft drink lids. You know that hole that you put the straw into in a soft drink lid? That hole is a lot bigger than it needs to be, but it can hold a small straw because of the little flaps. We planned to imitate this design in our modification to simply allow the Scribbler to wield multiple implements of varying sizes.

Kit Collage

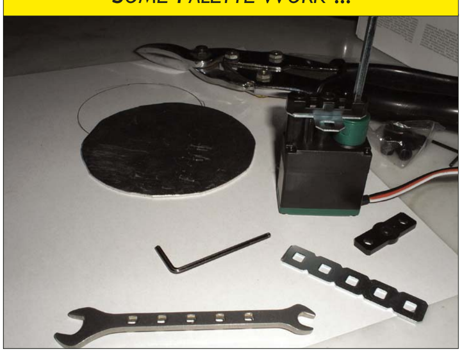
Materials selection is always interesting when hacking into a robot because, often, a disparate board of material fare needs to be bodged into a single working unit. One of the major pieces that we needed to find for our Scribbler hack was a motor for our mechanical palette. The motor wouldn't need to be terribly strong, but we wanted a compact and lightweight unit. A full rotation servo motor seemed like a good option, and there were plenty of robots lying around Robot Central (our garage) willing to donate one.

We were initially drawn to the servo motors from the Robovation kit — the practice bot from olden FIRST days. Now replaced by the VEX kit, the Robovation bot appeared to be a perfect donor for the Scribbler's mechanical palette modification. We had gotten a little ahead of ourselves when picking a motor, because we had overlooked one detail — the connector. We had not yet bothered to crack open the Scribbler to see what kind of bits could be bodged to its board, so we would have to reserve our judgment on the Robovation servo until after a bit of surgery.

The Scribbler is easily opened up by simply removing a few screws from the plastic body. To completely remove the top panel, the wires for the speaker need to be disconnected, but beyond the fact that the connectors are a snug fit, this is an easy operation. The Scribbler's PCB is already choc full of sensors, but the folks at Parallax and Element definitely had hacking in mind for this artistic bot. There is a free PWM port conveniently labeled "Hacker Port" — we were pretty sure that this would be the place to connect our motor.

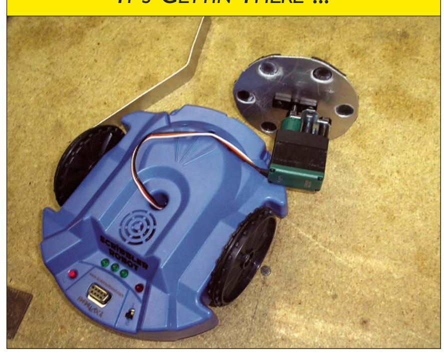
The problem was that the port didn't fit our servo — or rather, the connector on our servo motor couldn't connect to the Hacker Port. It would not be difficult to replace the connector on the servo, but there was an easy alternative to a fiddly connector

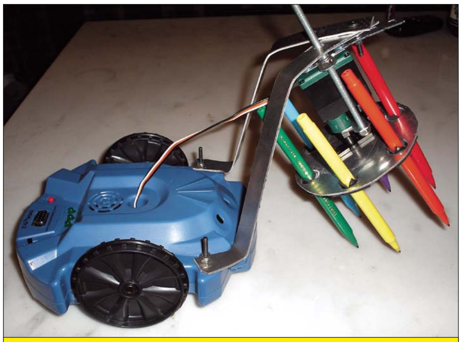
Some Palette Work ...



Yet, Some More Palette Work.

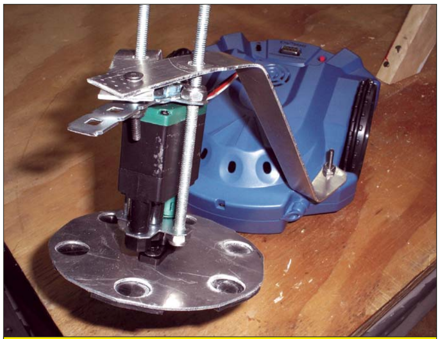
It's Gettin' There ...





Ta Da! It's Genius, We Know.

Notice the Difference in Pen Sizes.



The HACKED Scribbler.

transplant. We simply had to find a different useable servo.

Another possible motor donor was the VEX Robotics Design kit. The VEX servo motors had already served us well in competition, so we knew they were up to the task of spinning a mechanical palette (see the Twin Tweaks column in the July 2005 issue of *SERVO Magazine* for more details on our VEX experience). And it just so happened that the VEX servo motor had the right type of connector to fit into the Hacker Port. Thus, we had another instance where the VEX kit came to replace the Robovation kit — what a coincidence.

Color by Numbers

Now that we had all of the materials that we needed to give the Scribbler a mechanical palette — the VEX servo motor, pieces of rubber, and chunks of aluminum — we could dive into construction.

Nice Try, Scribbler!

A circular palette was easily jig-sawed out of aluminum, and the forever useful Unibit was a great way to drill out six media holders. With the Unibit, we could have just drilled out six different holes of varying sizes, but adding the rubber flanges to a set of all large holes gives the Scribbler the greatest versatility in its ability to hold multiple implements of varying sizes. For lack of a better adhesive, we attached our soda lid inspired rubber flanges with super glue.

While waiting for the glue to cure fully, we turned to the wiring. All that had to be done was plug in the servo motor to the Hacker Port, so wiring was not really a big deal. That's good news for hackers, though, because it is a testament to the hackability of the kit. The last minor issue that we had to deal with was routing the wires. The now defunct pen holder was a perfect path to the port for the PWM cable — with a little modification, that is. A small notch had to be cut into the bottom of the pen holder on the inside of the robot so the wires could find their way to the Hacker Port, but once the Scribbler was closed up, the injury to the bot's plastic body was invisible.

After the flanges were firmly adhered to the turntable, we could effectively assemble the mechanical palette. We connected the turntable directly to the VEX servo motor via an axle out of the VEX kit and another rubber flange. Now that the mechanical palette was assembled, it needed to be mounted.

To ensure that the Scribbler had a discriminating palette, we wanted to mount the turntable in such a way that only one of the six media would contact the paper at once. The easy solution was to mount the entire palette assembly at an angle. This way, only one implement would contact the paper while the rest were up in the air, awaiting their use. It was time for some metal sculpting.

The uneven shape of the Scribbler's plastic body didn't leave us many possible attachment points for the palette mount, but we only needed two. Our plan was to use two strips of 6061 aluminum as an adjustable mount — adjustable in the sense that 6061 aluminum is nicely bendable, even though we had to clamp it in a large green vise to do the adjusting.

After finessing the mount on the vise, we attached the palette assembly via some all-thread and some more pieces from the VEX kit. The masterpiece was mechanically finished — Scribbler now had the means to scribble with media ranging from paintbrushes to pencils, and in a multitude (well, six) of colors, if it so desired.

Okay, so the Scribbler wasn't quite up to the task of drawing an illuminated letter (yet), but it has the means now to do something like rudimentary pop art. Or it could dabble in impressionism. Whatever the case, the Scribbler now has the ability to put out some art that is at least better than those elephants can do. With good programming, that is.

Hack Writer

All that was left to do before the Scribbler was ready to do some serious art was a little writing — writing a pro-

gram, that is. To finely tune the action of the mechanical palette, we were sure that we would have to dive into the alien landscape of PBASIC, but thankfully, there are plenty of resources out there to help hackers who have more gusto than programming experience.

The official website of the Scribbler Robot (**scribblerrobot.com**) has plenty of ways for tinkerers with skill levels from novice to expert to amuse and educate themselves. There are plenty of programming guides available for free download on the website — novice users are encouraged to familiarize themselves with the ins and outs of the graphical user interface before moving onto the text environment of PBASIC. The website even has a caveat warning that the PBASIC environment is” Recommended for motivated roboticists ages 14 and up.”

Based on the fun time that we had with the Scribbler so far, we were confident that after using the Parallax kit, motivation would certainly not be a problem even for younger roboticists. The six page *Hacker’s Hints with Code* is a particularly accessible guide to the PBASIC syntax and commands specific to the Scribbler robot. Longer guides up to 300 and 500 pages detailing items like the BASIC Stamp are also available for those that really want to get an idea about what makes the Scribbler robot tick.

If your programming endeavors with the Scribbler still prove fruitless and you find yourself yearning for the good old days of the demo modes, fear not. Available both on the Scribbler robot’s website and on its CD is a function that restores the demo programs.

The only caveat that we have concerning programming the Scribbler robot is about downloading programs. The actual process of downloading a program is quite simple (just a couple clicks of the mouse), but the hardware involved might put off some users.

The Scribbler comes equipped with a serial port, and many computers (like ours) no longer have serial ports. This is only a minor snag, however, because a USB-to-serial adapter is an easy fix. We’ve had some bad experiences with unsupported adapters in the past, but once again the folks at Parallax have thought of everything — the website specifies exactly which adapters are compatible with the Scribbler.

Spanning the Spectrum

One of the great things about the Scribbler robot as an educational tool is that it uses technology that has applications far beyond the realm of a simple robotics kit. The BASIC Stamp is a Renaissance man of microcontrollers, with applications ranging from art exhibits to medical supplies to exploration of the oceans and of the solar system. Until recently, robots in the FIRST competition were even programmed in PBASIC, and we recall the words “BASIC Stamp microcontroller” uttered more than once at the autonomous submarine competition that we attended last year.

The mission of Parallax, Element, and Bueno Systems in creating Scribbler was to produce an accessible robotics kit that would appeal even to those that confessed to know nothing about robotics. After working with the Scribbler ourselves, we can confidently say that the mission has been accomplished.

The Scribbler artfully combines the simplicity of a fully assembled kit with the versatility of a developmental platform, and all of the resources available to Scribbler enthusiasts are not only designed to appeal to both novices and experts, but to turn those novices into experts, as well. **SV**

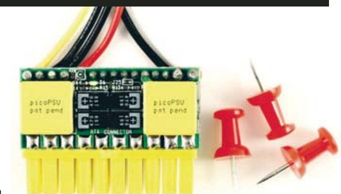
RECOMMENDED WEBSITES

For more information on the Scribbler robot, go to
www.scribblerrobot.com
www.parallax.com

NEW PRODUCTS

CONTROLLERS & PROCESSORS

Tiny picoPSU Power Supply



Mini-box.com — a division of Ituner Networks — has announced the picoPSU — the world's tiniest PC power supply. Measuring only 31 x 45 x 20 mm (about the size of two AA batteries), the picoPSU is 10 to 20 times smaller than standard PC power supplies. The tiny picoPSU delivers the same power as standard power supplies, enabling design of smaller PCs and smart appliances and giving devices such as robots less weight and mass to bear for longer periods of autonomous operation. The picoPSU is compatible with all Mini-ITX mainboards and standard motherboards.

Using patent pending HyperWatt™ technologies, the picoPSU-120 provides a cool, silent, 120 watts of power for small PC designs using a single 12V power source — an impressive amount of power relative to its tiny footprint. The picoPSU is fully ATX compliant, plugs directly into the ATX connector, eliminating a tangle of 20 unnecessary ATX wires, making it an excellent candidate for any silent, small design 12V DC-DC computer project. The picoPSU is also 1U Server Rack compliant, ideal for reducing noise and head in dense computing environments.

Inspired by VIA

VIA Technologies, Inc., is a leader in small form factor computing platforms with cool-running X86 CPUs and fully integrated chipsets. Ituner completes the picture with its cool, tiny picoPSU power supply, a crucial key component that delivers the same power as standard, bulky power supplies at a fraction of the size.

"VIA took the world by storm with low power, quiet, small computers and now mini-box.com is providing the missing link in small form factor computing: power. The picoPSU is a dream come true for PC modders, embedded applications, robotics, and industrial applications," said Andrei Bulucea, president of mini-box.com. "By reducing the overall size of the PSU by at least an order of magnitude, PC enclosures can now be manufactured smaller and require less internal cooling."

Cool Power

Operating at only 12V, the picoPSU DC-DC ATX power supply delivers 120 watts of power. picoPSU provides ample power (via ATX connector and HDD cable harness) for the CPU and an entire range of peripherals.

100% Silent

The picoPSU-120 mini PSU is a 100% silent DC-DC solution with no fans or noise, just power for small and silent PCs. picoPSU-120 is a fully compliant DC-DC ATX PSU. It can power VIA mini-ITX boards with C3 or C7 processors and lower power processors such as VIA's embedded CPUs.

Reduce Space

The picoPSU 12V DC-DC ATX converter was designed from ground up to fit Mini-ITX and small form factor computers, allowing enclosure designers to save space while not compromising power requirements.

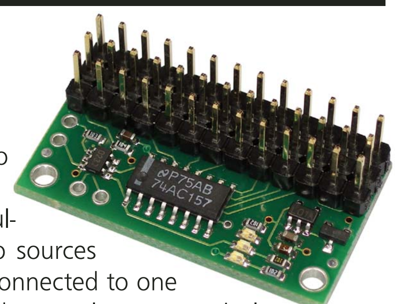
For further information, please contact:

Mini-box.com/ Ituner Networks

4031 Clipper Ct.
Fremont, CA 94538
Tel: **510•226•6033** Fax: **510•226•6032**
Email: **sales@mini-box.com**
Website: **www.mini-box.com**

MOTOR CONTROL

RC Servo Multiplexer



The Pololu RC servo multiplexer is a four-channel, 2-to-1 multiplexer that allows two sources of servo control to be connected to one set of servos or speed controls. In a typical application, the multiplexer is connected to a master radio control (RC) receiver and a slave microcontroller-based servo controller to allow mixed RC and autonomous operation of a robotic system. A fifth channel on the master input allows a human operator to remotely take over operation or to allow the autonomous system to control the robot.

The unit measures 1.3" x 0.725" and is available with or without connectors, starting at $19.95.

For further information, please contact:

Pololu Corporation

6000 S. Eastern Ave. Suite 12-D
Las Vegas, NV 89119
877•7•POLOLU or **702•262•6648**
Fax: **702•262•6894**
Email: **www@pololu.com**
Website: **www.pololu.com**

ServoCenter™ 3.1

Yost Engineering's ServoCenter™ 3.1 is an embedded controller that allows any device with a serial or USB port to control RC servo motors. Its unique design allows devices to easily control the seek position and seek speed of up to 16 connected servos independently and simultaneously. This independent control scheme allows one servo to move to a position slowly, while another is moving to a different position quickly, while yet another is moving to another position at a medium speed. This controller is especially useful for servo control applications in robotics, animatronics, motion control, automation, retail displays, and other areas where independent or coordinated fluid servo motion is desired.

The ServoCenter 3.1 controller can be programmed using a simple, raw serial protocol or can be programmed using the included ActiveX control or Win32 DLL. Example programs illustrating all programming methods are available on the included CD and the ServoCenter webpage.

For further information, please contact:

Yost Engineering Incorporated

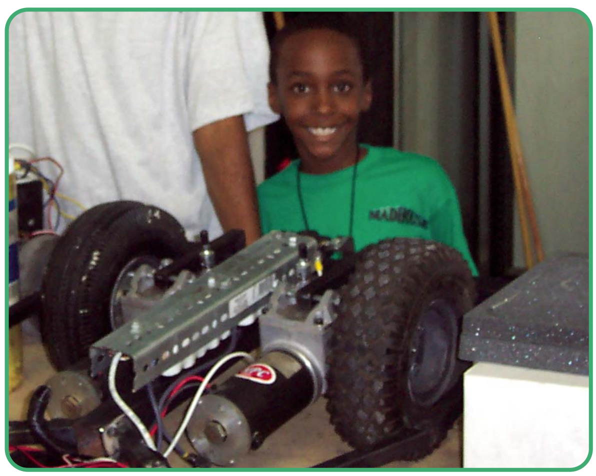
630 Second St.
Portsmouth, OH 45662
Tel: **888 • 395 • 9029**
Website: **www.YostEngineering.com**

MENAGERIE

ROBOT REBELLION 5.4

Photos provided by Bill Marsden

SWARC — The Southwestern Alliance for Robotic Combat (formed in 2000 by several north Texas robot enthusiasts) — recently hosted a "rebellion." Shown are some of the younger "rebels."



Send us a high-res picture of your robot with a few descriptive sentences and we'll make you famous. Well, mostly. menagerie@servomagazine.com

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

So you thought the DARPA Grand Challenge was a cool contest? Wait until you hear about the *egnellahC dnarG APRAD*. That's right — the DARPA Grand Challenge in reverse. It's also known by the more mundane name, Mobile Robotics Software Challenge. What is it? Well, in the Grand Challenge, each robot was issued a set of waypoints and had to use them to navigate from start to finish. That seemed too easy to Nelson Bridwell and the fine folks at the Portland Chapter of the IEEE Robotics and Automation Society.

So, they came up with something a bit harder. In their reverse Grand Challenge, they provide a robot that will take your computer and sensor package on a wild ride lasting about 10 minutes from start to finish. Your computer then has to provide the judges with a list of waypoints consisting of X, Y coordinates and orientation for the trip.

And no cheating with fancy navigation hardware like IMUs, gyroscopes, accelerometers, compasses or GPS. You have to figure it out in more interesting ways: optical flow sensors, vision recognition algorithms, or acoustic range-finders.

For those who like a challenge, they even provide some bonus points if you can do things such as determine whether — for any given point in the trajectory — the robot is within five seconds of a collision, or spotting special target objects revealed on the day of the event.

If this sounds like fun, you'd better get to work because the competition takes place on Saturday June 24, 2006 in Portland, OR. For all the details, check out the contest website at **www.mobilerobot.org/MRSC.htm**

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

March

10-11 **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana, Champaign, IL
This year's contest involves a 44' x 44' course in which robots from multiple teams will navigate ramps, overpasses, and teeter-totters in an attempt to collect and dispose of colored balls.
http://dc.cen.uiuc.edu

18 **Dallas Personal Robotics Group RoboRama 2006.a**
DPRG Lab, Garland, TX
The first of the DPRG's two annual RoboRama competitions. In case you haven't been to one lately, the contest events are now faster and more efficient. Check the website for a map to the contest location and the latest rules.
www.dprg.org/competitions/

18-19 **Manitoba Robot Games**
Winnipeg, Manitoba, Canada
Events may include both Japanese and Western style sumo, mini-tractor pull, and Atomic Hockey.
www.scmb.mb.ca

19-23 **APEC Micromouse Contest**
Dallas, TX
One of the best-known micromouse competitions in the United States. Expect to see some very advanced and fast micromouse robots.
www.apec-conf.org

21-23 **Singapore Robotic Games**
Republic of Singapore
Fourteen events including autonomous sumo, RC sumo, legged robot marathon, legged robot obstacle course, several levels of micromouse, wall climbers, pole balancers, and more.
http://guppy.mpe.nus.edu.sg/srg

April

1 **Penn State Abington Mini Grand Challenge**
Penn State Abington, Abington, PA
In this event, autonomous outdoor ground robots compete for a $400 prize by navigating around the campus, both on and off-road, avoiding obstacles.
www.ecsel.psu.edu/~avanzato/robots/contests/outdoor/contest05.htm

8-9 **Trinity College Fire Fighting Home Robot Contest**
Trinity College, Hartford, CT
The well-known championship event for fire fighting robots.
www.trincoll.edu/events/robot

21 **Carnegie Mellon Mobot Races**
CMU, Pittsburgh, PA
The traditional Mobot slalom and MoboJoust events.
www.cs.cmu.edu/~mobot

21-22 **National Robotics Challenge**
Veterans Memorial Coliseum, Marion, OH
In addition to sumo and maze solving events, this student competition includes two unusual ones: a robotic workcell event and a pick-and-place event.
www.nationalroboticschallenge.org

21-22 **RoboRodentia**
California Polytechnic State University, San Luis Obispo, CA
A micromouse-like maze navigation contest for autonomous robot mice. In addition to navigating the maze, robots must pick up balls and place them in a nest.
www.ceng-web.calpoly.edu/openhouse/saturday.php

22 **UC Davis Picnic Day MicroMouse Contest**
University of California, Davis Campus, CA
Standard micromouse contest.
www.ece.ucdavis.edu/umouse

23 **Dominican Republic National Robots Competition**
Santo Domingo, Dominican Republic
Robots must locate radio and light-emitting beacons. Robot builders must type in the URL. I'm not sure which task is harder.
www.indotel.gov.do/(uyxlzr55lghapq45ntdf2245)/concursos_article.aspx?article=297

27-29 **FIRST Robotics Competition**
Atlanta, GA
National Championship for the regional FIRST winners.
www.usfirst.org/

27-28 **HISPABOT & Alcabot**
University of Alcala, Madrid, Spain
Sumo, maze solving, and Alcafutbol (soccer).
www.depeca.uah.es/alcabot/hispabot2006/

28 **RobotRacing**
University of Waterloo, Ontario, Canada
Autonomous cars must race head-to-head on outdoor courses. A two car drag race on a 20 meter straight course is followed by a multi-car, multi-lap race on a 150 meter circuit course. The circuit course is bounded by orange cones and GPS waypoints are provided.
www.robotracing.org/

29 **The Tech Museum of Innovation's Annual Tech Challenge**
San Jose Civic Auditorium, San Jose, CA
A different robot challenge is designed each year. Check the rules on the website for the details of this year's challenge.
http://techchallenge.thetech.org/

May

10 **Micro-Rato**
University of Aveiro, Aveiro, Portugal
Micro-Rat competition (similar to micro-mouse, but larger).
http://microrato.ua.pt/

13 **Atlanta Robot Rally**
Souther Polytechnic, Marietta, GA
Open Contest — contestant chooses his own goal for robot. Vacuum Contest — autonomous household vacuuming contest Mini-Sumo.
www.botlanta.org/Rally/

13 **RoboFest**
Lawrence Technological University, Southfield, MI
The RoboFest includes many events, such as LEGO robot competition, LEGO robot exhibition, advanced robot competition, and advanced robot exhibition.
http://robofest.net/

Fighting Robots Return to Daytona Beach

Battle Beach, LLC, is pleased to announce that their next robot combat event — Battle Beach 4 — will be held on April 8th and 9th, 2006. This is the Robot Fighting League's Southeast Championships and an RFL National Championship Qualifier. Battle Beach 4 will feature remote controlled fighting machines in ten weight classes ranging from 5 ounces to 340 pounds. Instead of lounging on the sands of Daytona Beach, these flippers, full-body spinners, and flame throwers built by teams from as far away as Wisconsin and Las Vegas will spend their Spring Break in a steel and polycarbonate box wreaking havoc on each other. Previous Battle Beach events have featured hundreds of these unique, exciting machines, some costing as much as $20,000 and this next event is shaping up to be a real eye opener.

More information about Battle Beach and Battle Beach LITE events may be found at www.battlebeach.com and information on the Robot Fighting League National Championships may be found at www.botleague.com

GIVE YOUR Microcontroller MORE DRIVE

by Tom Napier

An addressable latch lets microcontrollers set and clear individual output bits but also generate short pulses.

If there's one thing I have learned in 10 years of developing devices based on small microcontrollers, it is that they never have quite enough I/O port pins.

One solution is to add an external latch or demultiplexer chips. This approach was described by Tom Dickens in an article in the October 2004 *Nuts & Volts*. Demultiplexers direct a negative pulse to one — but only one — of many output pins. You can't control the state of more than one pin simultaneously in order to send, for example, pulse width modulated signals to several servos or serial data to a terminal.

One alternative is a four-, six-, or eight-bit wide register chip. When clocked, this acquires and retains the applied parallel input values. It has the disadvantage that all its outputs must be set at the same time, even those that you don't want to change. To modify only one or two bits, you need to keep a copy of the current output state in an on-chip register, change it, then write the result to a controller port before clocking the register.

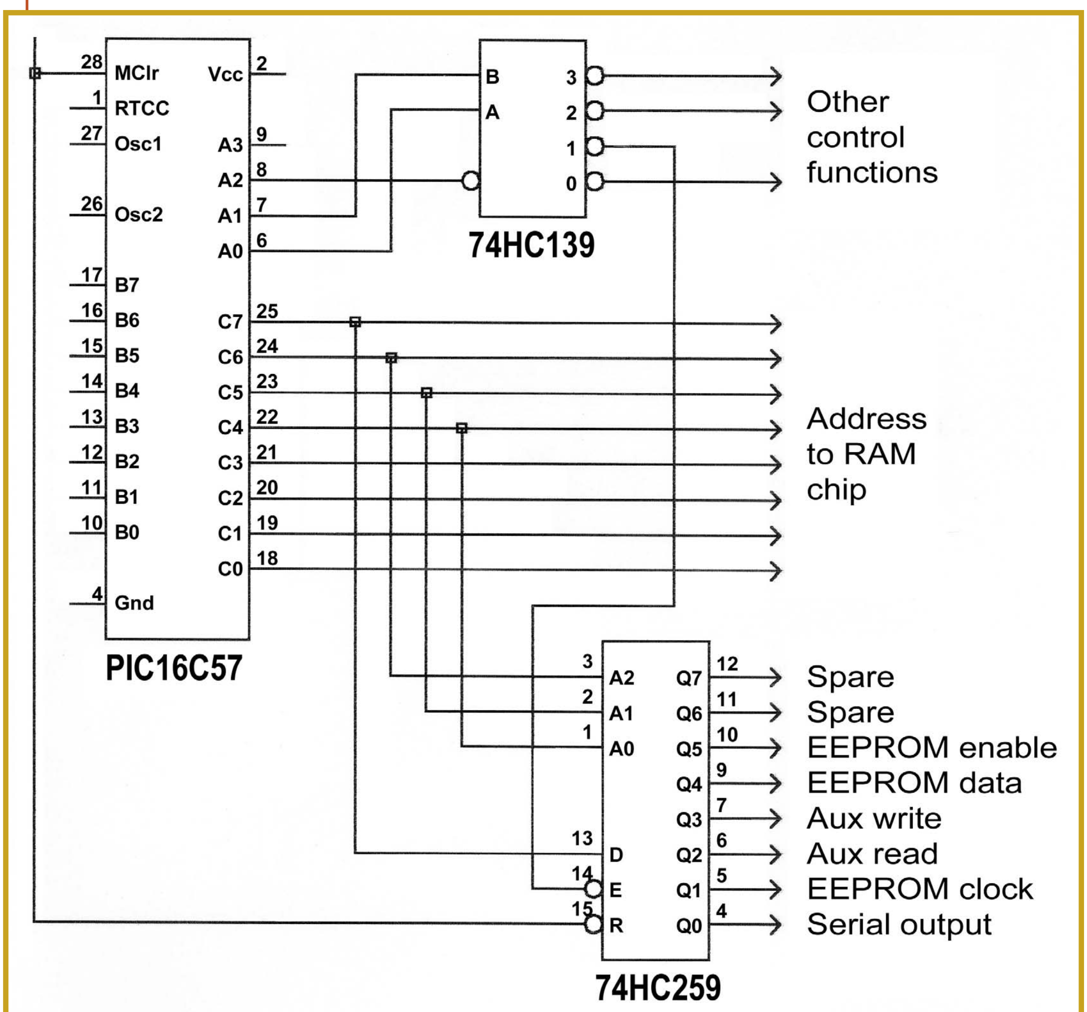


FIGURE 1. This typical application shows the pin-out of the 74HC259. Here it shares PIC pins with other functions to supply eight independent outputs.

Pulse or Latch

There's a more flexible way of expanding the output capabilities of a microcontroller: the 74HC259 eight-bit addressable latch. This chip gives you the best of both worlds. It can toggle any one of its eight output bits up and down,

but it can also set any output to a new value that will be retained indefinitely. As Figure 1 shows, I've used this chip to do both jobs in a single product. For example, one output bit is set to a new value every 104 µS. This generates 9600 baud serial data. Another bit is set low and high with successive controller instructions, making a 200 nS active-low pulse to latch eight data bits into a register. Two more bits drive the 400 kbps data and clock buses of a serial EEPROM. Another bit multiplexes the EEPROM's data bus to a microcontroller input pin during reading.

As Figure 1 shows, the 74HC259 has six input pins that control what happens to its eight output pins. Three address inputs select one of the eight output bits; the data input pin determines whether that output pin is set high or low. The remaining two inputs are the chip enable and reset. These can be combined to perform several functions.

An Active Enable

If the enable pin is high, the chip ignores all inputs except the reset. All eight outputs retain their previous state. Lowering the reset pin clears all eight output latches. This pin would normally be connected to the microcontroller's master reset input to clear all the outputs at power on. The controller's start-up code can then set any outputs that should be active low.

When the enable pin goes low, the state of the data input is loaded into whichever of the output bits is currently addressed. As long as the chip is enabled, the output latch is transparent, that is, its output state follows the data input. This means, for example, that a PIC running with a 20 MHz clock can generate bursts of 2.5 MHz square waves to clock an eight-bit shift register simply by alternately setting the data input high and low.

When the enable goes high, the output pin retains its last state. The address can then be changed to drive a different output pin. It's not recommended to change the address while the enable pin is low; this might lead to address glitches that change the state of the wrong bit. You can get away with cycling through all eight addresses with the enable held low as a quick way to set all the outputs high or low.

To generate signals that rarely change, send a bit address and set the data pin to the appropriate value. Sending a short negative pulse to the enable input latches the selected data bit. The next time the output must change, the process is repeated. All eight output bits can be driven this way with different frequencies and on-off ratios. Between changes, the controller can be doing other things.

One odd feature of the 74HC259 is that its enable pin overrides the reset pin. This means that if the data pin is high, the enable low, and the reset low, the 74HC259 behaves as an active-high eight-way demultiplexer. All the outputs are low except the one that is addressed. This gives you a positive-going equivalent of the 74HC138, should you need it.

Sharing Ports

You might think that if you have to drive five or six input pins to control eight output pins, you haven't gained much. But, in fact, you have. In many cases, the 74HC259 can share controller outputs with other functions. Only the enable pin needs to be a dedicated output, and even it can be a decoded state of other pins.

In the application partially shown in Figure 1, a PIC16C57 drives RAM chips with an eight-bit bi-directional data bus (Port B) and eight-bit multiplexed address bus (Port C). The 74HC259 can be addressed whenever the RAM is not enabled. Four bits of the address bus are connected to the address and data bits of the 74HC259. Its reset is low only at power on. Only its enable pin is unique to the 74HC259 and even that is driven by a decoder chip, which is there for another reason: one of its outputs latches the high RAM address into a page register.

Once you have one 74HC259, it's easy to add more because they can share a common address and data bus. Only their enable pins need be driven separately by, for example, half a 74HC139 decoder chip. Connecting four 74HC259s and a decoder to six port pins expands a processor's drive capacity to 32 outputs. Each pin can be set and cleared when you wish, but can also provide short pulses as needed. The 74HC259 is indeed a versatile output driver. SV

RADIO COMMUNICATION BASICS FOR ROBOTS

Or, How to Reliably Break the Tether!

by Steven Schmitt

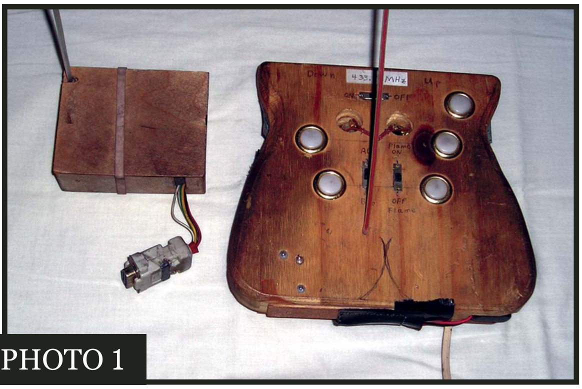


PHOTO 1

PHOTO 2

PHOTO 3

A custom radio control using serial link modules solved a need for a low cost radio control system that used relays to control the drive motors and weapon systems. These serial link modules are available in both AM and FM types from several sources. While RF systems require special care in their construction, any hobbyist should be able to design and build a radio system using serial link modules with the usual hobbyist tools.

The FCC (Federal Communications Commission) has allocated the frequency spectrum to many different uses and is very particular about certifying radio products for commercial use. However, part 15.23 of the code states that hobbyists may have up to five unlicensed, low power transmitters. Refer to the FCC regulations for the permitted frequencies and power levels.

Our custom radio was built using FM serial link modules and Atmel microprocessors. The transmitter and receiver are shown in Photo 1. Photo 2 shows a group of four robots with their radio controllers. We use these in school demonstrations where students take turns playing bot-ball. The single transmitter has four control pads, one for each bot. The transmitter sends out control commands by using a bot address appended to each command.

A close-up of one of the bots is shown in Photo 3. The receiver, with the serial link module and helical antenna, is in the left compartment. The motors are from an automotive power mirror and are powered by four nine-volt batteries.

Selecting a Frequency

The common frequencies for serial link modules are 315 MHz, 418 MHz, and 433.92 MHz. These frequencies have many common uses, but by limiting the range and the operating time to a fraction of a second, they can accommodate the demand.

The FCC rules require that part 15 radio systems cannot cause harmful interference and must be able to accept interference. This implies that the system should have good error checking and failsafe.

Mouser Electronics carries serial link modules priced at

> **"When purchasing a commercial RC radio system for model airplanes or model cars, the main types of radio systems may be referred to as AM, FM, or PCM."**

$3.99 for the transmitters and $4.26 and $4.99 for the receivers. The transmitter/receiver pairs are available in 315, 418, and 433.92 MHz versions. The receivers are simple AM, super-regenerative designs with a bandwidth of 3 MHz, and the transmitters are single transistor oscillators. There is also a more expensive super-het receiver with a bandwidth of 150 kHz. These are all on/off keying (OOK) systems.

Modulation Basics

For data transmission, any serial link module will have OOK, amplitude shift keying (ASK), frequency shift keying (FSK), or phase shift keying (PSK) modulation. FSK (FM) and OOK (AM) are the two main types, while ASK and PSK are seldom used.

When purchasing a commercial RC radio system for model airplanes or model cars, the main types of radio systems may be referred to as AM, FM, or PCM. Another term that is occasionally used is PPM. All of these radios are either AM or FM. The terms pulse code modulation (PCM) and pulse positional modulation (PPM) are ways to encode servo information and are not modulation techniques. PCM or PPM encoding can both be sent over either AM or FM. Generally, PCM will be an FM radio, while PPM could be either AM or FM.

PPM and PCM are how the data is encoded and sent to the individual servos in the model. The signal required by a servo consists of 60 pulses per second and are 1.5 ms for centered, 1.0 ms for full left, and 2.0 ms for full right.

The PPM transmitter sends out 60 frames per second. Each frame consists of a long header followed by a series of pulses; one pulse for each channel or servo. The receiver simply routes the pulses to the servos. The first pulse goes to the first servo, the second pulses goes to the second servo, and so on. The length of each pulse directs the servo.

The PCM transmitter has a microprocessor to encode the control signals and sends out a byte of data for each channel or servo. The microprocessor in the receiver decodes the byte of data and generates the control pulses for each servo. Generally, 0 means full left, 128 centered, and 255 full right. In addition to the data bytes for the servos, there is a header and check bytes or possibly error correction bytes. High-end radios may use two bytes for each servo for greater resolution.

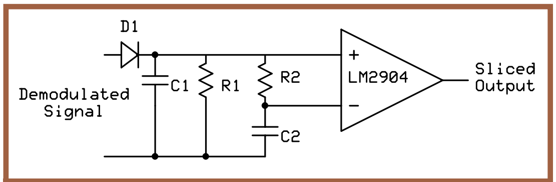


FIGURE 1. Typical Data Slicer.

Noise Rejection

When a PPM receiver gets a noise pulse, the sequence of servo pulses is confused and everything after the noise pulse will be incorrect. This will cause the servos to chatter or if there are electronic speed controllers (ESCs), the motors will twitch. If the next frame is clean, the problem may not be noticed.

When a PCM receiver gets a noise pulse, the check bytes will be incorrect and the frame will be discarded. There should be no chatter or twitching when using a PCM radio. If enough frames are bad, the PCM receiver will go to failsafe mode and return all controls to a predetermined neutral position. For a robot, the failsafe mode should be enough to bring everything to a stop.

Noise is very visible when using an AM or FM PPM radio, while a PCM radio will mask the noise and appear to be working much better. In general, there is very little difference in range between an FM or PCM radio as they are both FM radios. The cheaper AM radios will most likely be wide band super-regens and have less range and more noise.

Serial link modules will generally be used with a microprocessor and there is no need to use standard servo signals. Any type of protocol can be used. Since the FCC requires the receiver to accept any interference, some type of error checking is required making it a PCM system.

Noise vs. Interference

Interference and noise are two concerns with any radio link. Interference is when another transmitter is operating at a frequency that is detected by the receiver. In addition to the operating frequency, it can also be a multiple of the IF frequency added or subtracted from the operating frequency or a very powerful jamming transmitter. Fortunately, interference does not occur as often as many people believe.

Receiver noise is a much larger problem and is sometimes referred to as interference. The two sources of noise are received and internal. Received noise can be anything from lighting to electrical sparking. There is very little that can be done to eliminate external noise other than to find the source and fix it there. However, at these higher frequencies there are few noise sources.

In addition to the received noise spikes, there is a low level of internal background noise that is always present and limits the range. Since the amount of noise detected is proportional to the receiver bandwidth, a super-het receiver with a narrow bandwidth will have a better range than a wideband super-regen receiver. Most low cost AM receivers are super-regen receivers and all FM receivers will be super-hets. A good super-het AM radio will have about the same range and reliability as an FM radio. One application note from RFM reports that ASK is even better than FSK and that OOK is very close to FSK.

Power In Data

The serial link module consists of a receiver circuit and a data slicer. Figure 1 shows a typical data slicer used for either OOK or FSK. The R2/C2 filter stores the average level of the demodulated signal which is used as the reference for the comparator. When the received signal is very weak, the comparator will be susceptible to noise. Also, when there is no RF input signal, the two comparator inputs will be at the same level and the comparator will output high frequency hash which could cause unexpected robot movement. A failsafe is required to shut everything down or the transmitter must be on when the receiver is on. Hence the adage, turn the transmitter on first and

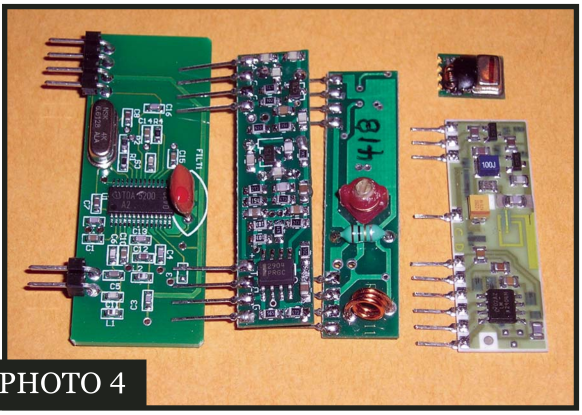


PHOTO 4

Equation 1
Antenna length in inches = 2808 / Frequency in MHz

off last. The comparator could be biased to have a guaranteed output without RF input, but the biasing would reduce the sensitivity by several db, which would cut the range in half.

The data slicer operates most efficiently when the reference level is halfway between a 1 and a 0. This means the data stream should have about the same number of 1s as 0s. Any other data stream will reduce the range and reliability.

One way to have equal numbers of 1s and 0s is to use Manchester encoding where each 1 bit is sent as 10 and each 0 bit is sent as 01. For example, to send a data byte of 10100101 with a UART using Manchester encoding, the bit stream would be 0100110011001100 1101. The extra bits are the start and stop bits inserted by the UART.

The system can send continuous data or data packets. In either case, there has to be a way for the receiving circuits to synchronize the data. With data packets, the data slicer needs a string of alternating 1s and 0s to set the reference level before data can be decoded. With a continuous data stream, the slicer always has the reference level set. In either case, the UART needs to see a clear start bit to begin. One way is to send a byte of 0s followed by a byte of 1s. With continuous data, there has to be a few bytes of 0s or 1s for the UART to stay synced.

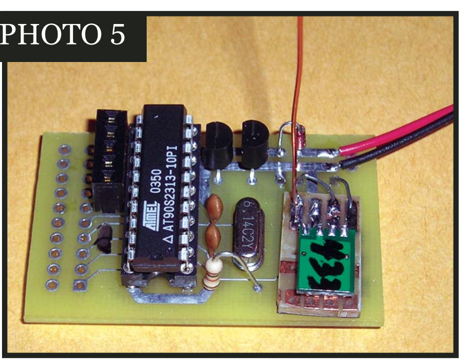


PHOTO 5

Reliable Encoding

A good encoding scheme for data packets is shown in the referenced application note at the end of this article from Radiotronix. The encoding scheme replaces each four bit nibble with a six bit nibble. Each six bit nibble has the same number of 1s and 0s. The Radiotronix website also has several other application notes including schematics for transmitters and receivers using the serial link modules.

Another way to send data is to send each byte twice: first normal, then second inverted. This provides redundancy, error checking, and sends an equal number of 1s and 0s. It is also the easiest method when using a microprocessor and UART. Using this method will cause the reference level at the data slicer to wander a bit, reducing the range by a small amount.

Understanding Range Issues

Given a transmitter/receiver combination, there are three ranges of operation. The near range is where the signal-to-noise ratio is high and the receiver sees a clean signal. Next is a midrange where the signal-to-noise ratio is low and the receiver generates some noise. Finally, the far range is where the receiver cannot detect any signal above noise. Given all the things that can degrade a radio signal — such as metal shielding, multi-path fading, etc. — the far range, using the same equipment, may be several hundred feet away or just a few feet. Normally, radio signals drop off at the square of the distance, but inside a building it can be as bad as the fifth power of the distance or even worst. Lower frequency radios with long wave lengths have severe range problems in metal buildings.

While there are technically many ways of improving range, most are not practical for simple systems. About the only easy way to improve range is to use the best antennas and use a superhet receiver with a good signal-to-noise ratio. Also matching the frequency with the intended environment helps; 900 MHz works much better inside a battle cage than 75 MHz.

Error correction is of limited help in extending range as the midrange is typically small and the number of error bits gets to be very large. Redundancy and short frames are much more effective. However, with large redundancy it is easy to implement some form of error correction. A simple example is sending the same byte four times and using the data if any three of the four bytes match.

Bunnies or Duckies?

There are a vast number of antenna designs, but for simple radio systems only three are practical. The simplest antenna is a piece of wire of some length, taped to the side of the robot. It should be under a 1/4 wavelength and in about the same orientation as the transmitter antenna. The second design is the 1/4 wavelength monopole antenna, and the last design is the helical antenna.

Equation 1 can be used to calculate the length of a monopole antenna. The actual length is a bit shorter than 1/4 wavelength due to the physical wire size and fringing effects. Helical antennas are mostly designed empirically; make a long coil and shorten it for maximum range. One design for 433 MHz is a coil of 17 turns over a 5 mm core, 34 mm long. Both the diameter and length have to be small compared to a wavelength, otherwise it becomes a loop antenna. The field from a helical antenna is circular polarized so all antennas have to be wound the same direction.

Both the 1/4 wavelength monopole and helical antennas have a similar, broad directional characteristic. Other

types such as the 5/8 monopole have better range, but the antenna has to be oriented correctly to get the advantage of the range. Since handheld transmitters get tipped every which way, they work best with 1/4 wavelength monopole or helical antennas.

When attaching the antenna to either the transmitter or receiver, the connecting wire has to be short compared to the wavelength. Long connections require 50 ohm transmission lines. One reference indicates anything longer than 1/8" needs to be 50 ohm cable. Since a 1/4 wave monopole antenna has an impedance of 36 ohms, matching circuits would be needed to connect it to a 50 ohm cable.

Putting It All Together

I have used both AM and FM serial link modules for controlling combat robots. They appear to be as reliable as any other radio system. To evaluate the serial link modules from Mouser Electronics, I ordered three sets of transmitter/receiver pairs at 315, 418, and 433.92 MHz. With these modules, I made some range measurements to get an idea of how well they work.

Photo 4 shows the receivers and transmitters. On the left side is the 433 MHz super-het AM receiver. Note the crystal for generating the IF frequency and the ceramic IF filter. The two modules in the center are the front and back sides of the Mouser AM receivers. The radio portion fills the top of the circuit board and at the bottom is the LM2904 comparator for the data slicer. On the right is an AM receiver from OKW Electronics. It has a very clean ceramic circuit board with thick film resistors and etched inductors, but is very similar to the Mouser radios. It has the usual two transistor AM section and the LM2904 data slicer at the bottom. In the top right is the Mouser transmitter, a very small four pin device.

I attached the serial link modules to my circuit boards that were laid out for the FM modules from OKW Electronics. To connect the Mouser parts I had to use jumper wires to a tiny daughter board. My circuit boards do not have any ground or voltage planes which is not good for proper operation. The transmitter board is shown in Photo 5 and the receiver board is in Photo 6. The transmitter board has nine inputs for attaching nine switches to control the relay attached to the receiver. The receiver board has space for seven relay drivers to control motors of any size. The five pin sockets on both the transmitter and receiver boards are for in-circuit programming of the micros. The antennas are the wires extending out the tops of the photos.

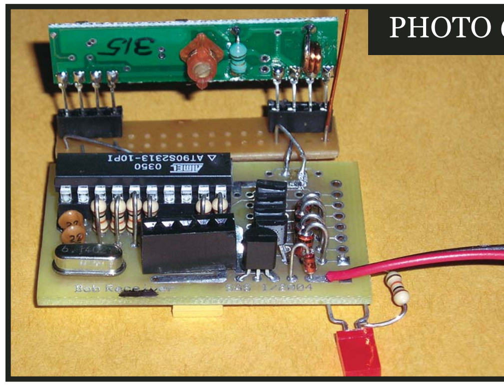


PHOTO 6

Software Considerations

The encoding was the simple method of sending each command byte four times and using the command if any three of the four bytes were the same. Bytes two and four were inverted to equalize the number of 1s and 0s. Between the four command bytes was a byte of 0s for the UART to sync on. Whenever a good command was received, the deadman timer was reset. If a good command was not received within 1/10 second, the deadman timer reset everything for a foolproof failsafe. While this is not an optimum encoding method, it is good enough to make the system work.

I programmed the transmitter to send out a command to blink an LED attached to the receiver. The transmitter was placed in the front window of my home and I walked down the street until the LED attached to the receiver quit blinking in sync with the transmitter. The LED would try to blink at further distances but the failsafe was resetting the receiver.

At the maximum range, the LED was still blinking in sync with the transmitter, but there probably was a large number of errors which were being masked by the microprocessor. The errors were not visible until the failsafe reset the receiver and turned off the LED.

At maximum range, there were several houses between the transmitter and receiver. This was not an open field measurement and should not be interpreted as the capabilities of the modules. It is just an indication of what to expect. The results are shown in Figure 2. For reliable operation, the usable range is about 1/3 or 1/4 of the maximum range.

Conclusions

The results only apply to my circuit boards and encoding methods. Without the proper measurement equipment, it is impossible to know how good my antennas were or how well the modules were working. My objective was to see how well the modules would work in a hobbyist environment without any optimization.

The transmitters were very sensitive to the presence of a ground plane and would quit oscillating if a metal cookie sheet was placed under the circuit board. In fact, the transmitters would sometimes quit if I put my hand under the circuit board. These are simple, one transistor circuits designed to operate with a 50 ohm load, so the antenna is a major part of the circuit and any conductors near the transmitter affect it.

Given that the monopole antenna

FIGURE 2. Results of range tests.

- With a monopole antenna at both ends, the range was about 500' for the three frequencies.
- With a transmitter ground plane and shortened monopole, the range was about 500' for the three frequencies.
- With one monopole and one helical antenna the range was about 350'. It did not matter which end had the helical antenna.
- With a helical antenna at both ends, the range was about 300'.
- With monopole antennas at both ends, the super-het receiver had a range of 800'. It also required a large ground plane under the receiver.
- With no ground plane, the super-het receiver had a range of 20'.

Resources

Mouser Electronics
www.mouser.com

Radiotronix — manufacturer of radio products
www.radiotronix.com

OKW Electronics — manufacturer of radio products
www.okwelectronics.com

RF Solutions — manufacturer of radio products
www.rfsolutions.co.uk

Schematics for radio systems
www.radiotronix.com/downloads/

Encoding techniques
www.radiotronix.com/datasheets/

Excellent application notes
www.rfm.com/corp/apnotes.htm

Many excellent newsletters
www.rheintech.com/mpnews.shtml

has an impedance of 36 ohms and the transmitter requires a 50 ohm load, I tried different length antennas to fine-tune it. Starting with a long antenna, I cut off 1/2" lengths to see the effects. At either 7.5" or 5.5", the transmitter worked with a ground plane and was immune to where my hands were placed.

As expected, the super-het receiver had the best range and is about as good as an FM receiver. However, the super-het receiver was very sensitive to the antenna and required a ground plane. The super-regen receivers worked the same with or without a ground plane.

Increasing the transmitter voltage from 3.3 volts to nine volts changed the RF output power and doubled the range. The super-regen receivers then had a range of 800' and the super-het range was farther than I wanted to walk.

For all of these tests, the transmitter was about 8' above ground and the receiver was at handheld height. In normal usage, the transmitter would be at handheld height and the receiver would be near ground. To test the effect of height, I placed the transmitter on the ground and the receiver at handheld height. The result was that the range was cut in half. This means the reliable working range for these AM systems is about 50 to 75 feet. Since all link modules use the same FCC mandated power levels, they will all have about the same range.

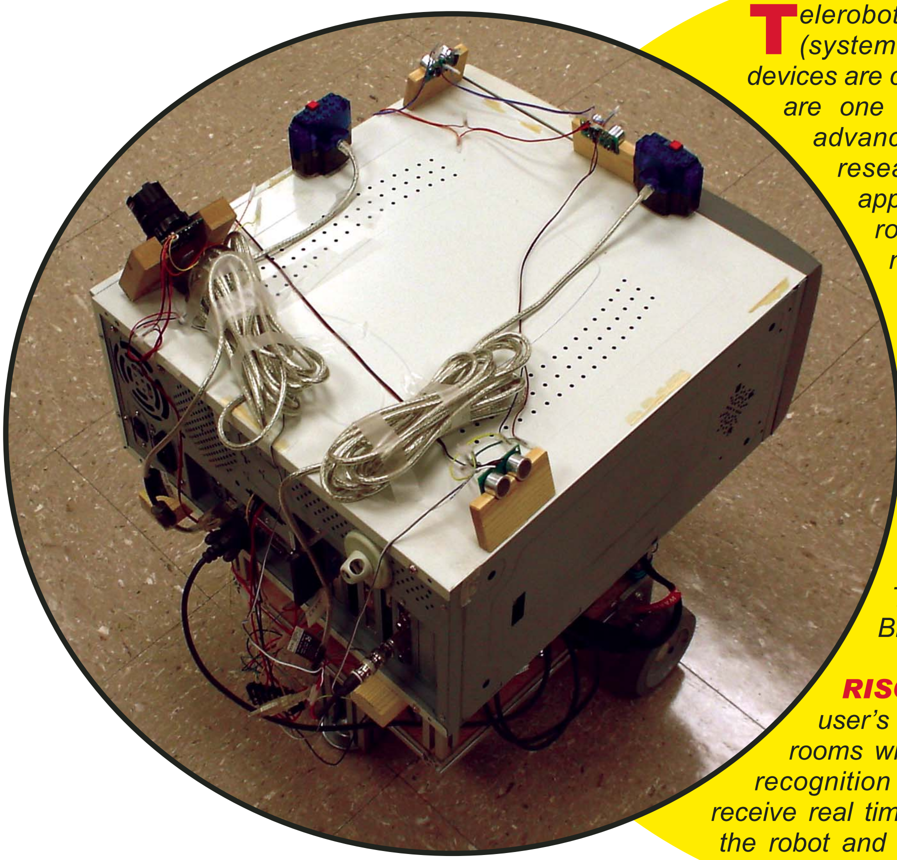
The interference test I ran involved using the 315 MHz transmitter and my car door's remote entry transmitter. The remote entry system was totally blocked by the 315 MHz transmitter. I did not test the size of the interference area.

Since these frequencies will interfere with your neighbors, it's best to keep the transmitter power as low as possible and use them for short bursts. The microprocessor should be programmed to turn off the transmitter after a few minutes in case the transmitter is inadvertently left on. With an OOK system, the transmitter can be turned off by setting the data line low. Any good FM transmitter will have an enable input to turn off the transmitter.

If continuous transmission is required, do not use these frequencies. There are other frequencies allocated for continuous transmission. **SV**

RISCBOT

by Tarek Sobh, Sarosh Patel, and Rajeev Sanyal of RISC, Laboratory University of Bridgeport, CT



Telerobotic systems (systems in which robotic devices are controlled from a distance) are one of the modern marvels of advanced interdisciplinary scientific research and development. The application spectrum of these mobile robots ranges from autonomous museum guides to space exploration and remote surveillance, and from hazardous material handling to running errands. With the advent of the Internet, telerobotics has received a major boost.

***RISCBOT**'s design commenced with a vision to provide a simple, fast, and reliable telerobotic system. Today, it can be commanded via the Internet to move to particular offices in the Engineering Technology Building of the University of Bridgeport, CT.*

***RISCBOT** localizes and fulfills online user's requests of navigating to desired rooms with a visual door ID character recognition algorithm. Online users receive real time video feedback from the robot and can also view the robot position.*

This article describes RISCBOT's architecture, its modules, applications, and, finally, future proposed enhancements.

Mechanical Construction

RISCBOT is the experimental telerobotic system we built. It is modular in structure. Pro Engineer (ProE) was used to design and visualize various configurations of the robot in the initial design phase. The best design was selected based on stability criterions, speed, floor clearance, turning radius, and appearance. RISCBOT has been built with T-slotted aluminum extrusion rods as it provides the liberty of altering the design and making changes with ease. Figures 1 and 2 show views of RISCBOT during the initial ProE design phase.

RISCBOT navigates using a front end differential drive, implemented with two 4" wheels (see Figure 3) and a rear caster wheel for support, as shown in Figure 4. Two 12V DC servomotors (Pittman) drive the wheels. An ATM103 MCU controls the ultrasonic sensors and the two motors. The PC sends commands to the MCU through the serial port. Atmel's ATM 103 (see Figure 5) is an eight-bit RISC MCU being used to control the DC motors and interface with the PC board (serially) and the ultrasonic sensors.

The data from the ultrasonic sensors is monitored in an interrupt based embedded C code on the ATM 103. The main program running in the eight-bit CPU for controlling the motors depends on the instructions sent serially from the PC. A nine-volt Panasonic battery (see Figure 7) with an inverter (see Figure 8) serves as the power house for the robot.

The PC cabinet housing the WLAN card and an NM 6403 DSP board (see Figure 6) is mounted on top of the base. Three ultrasonic sensors, two Logitech cameras, and an NTSC camera are mounted on the PC cabinet. The NM6403 DSP board performs a visual recognition algorithm when signaled by the PC. The NM6403 is a high performance dual core micro-

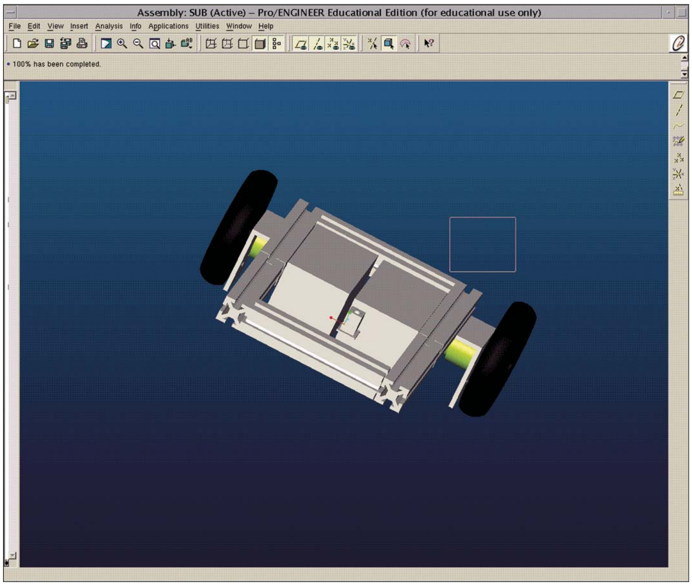


Figure 1. Top view of RISCBOT.

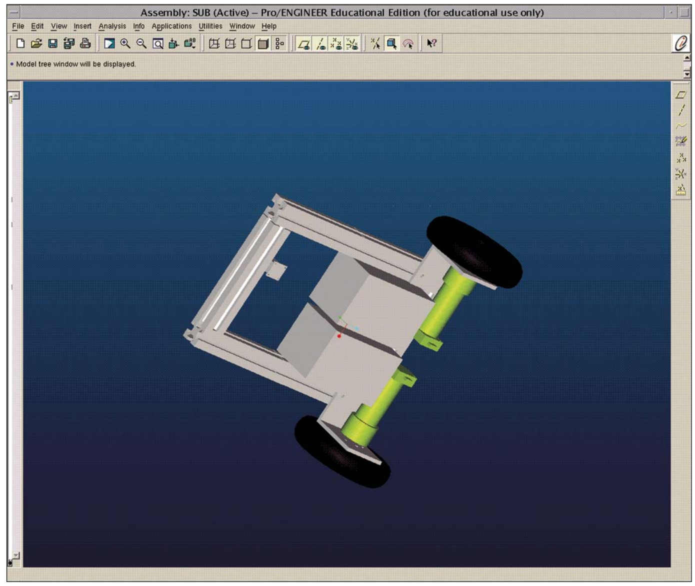


Figure 2. Bottom view of RISCBOT.

processor with a combination of VLIW/SIMD architectures.

The architecture includes two main units: a 32-bit RISC Core and a 64-bit VECTOR co-processor to support vector operations with elements of variable bit length. The NM6403 based reconfigurable high-performance multi-DSP development set consists of four NM6403 DSP processors, 16 MB asynchronous SRAM, 16 MB SDRAM, 1 MB Flash, NTSC, and PAL video decoder and PCI host interface. The PC and the NM6403 share 4 MB SRAM.

Navigation Module

The robot waits until it receives information from the server. Once it receives a command from the server, it starts searching for the requested room. The robot navigates along the wall to the left side of the corridor. With the help of the onboard ultrasonic sensors (see Figure 10), the robot maintains a safe distance of 45-50 cm from the wall. If the robot gets closer to the wall, it turns right, if it gets farther away, it turns to the left, and if the distance from the wall is within 45-50 cm, the robot continues to move straight.

If the robot encounters a wall right in front of it (example, at corners), it takes a right turn. The image processing program checks for doors continuously. Once the program detects a door, it signals the NM6403 DSP board to check for the room ID. If the room ID matches the requested ID, the robot stops. If not, the robot continues moving until it finds the desired room. Figure 9 shows the control flow diagram for the navigation module.

Image Processing

The door recognition algorithm is computationally fast, so that doors can be recognized in real time and appropriate commands can be sent to the Navigation module to stop the robot in front of the desired door. Our algorithm employs edge detection to differentiate between the wall and the door. We used various filtering techniques for edge detection. The best results were obtained using a Gaussian — a Laplacian filter — also commonly known as the Log filter. This module is programmed in MATLAB.

Images from the camera are captured on the run from the mounted webcams (see Figure 11). Figures 12-16 show a set of images captured by the camera and Figures 17-21 show

RISC Laboratory
University of Bridgeport, CT

The Interdisciplinary RISC lab resides in the Computer science and Engineering department at the University of Bridgeport, CT. It was formed in 1995 by its founder and coordinator, Professor Tarek Sobh, in order to do research in a variety of robotics-related fields, and as a step towards the development of commercially-applicable projects.

Their research interests include reverse engineering and industrial inspection, CAD/CAM and active sensing under uncertainty, robots and electromechanical systems prototyping, sensor-based distributed control schemes, unifying tolerances across sensing, design and manufacturing, hybrid and discrete event control, modeling, and applications, mobile robotic manipulation, developing theoretical and experimental tools to aid performing adaptive goal-directed robotic sensing for modeling, observing and controlling interactive agents in unstructured environments.

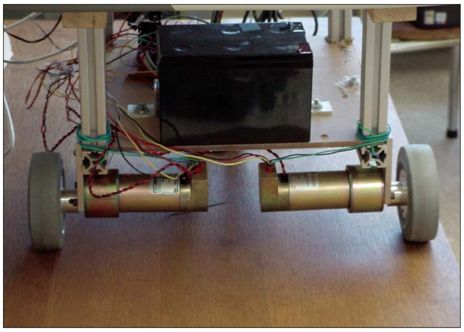


Figure 3. Front end differential drive.

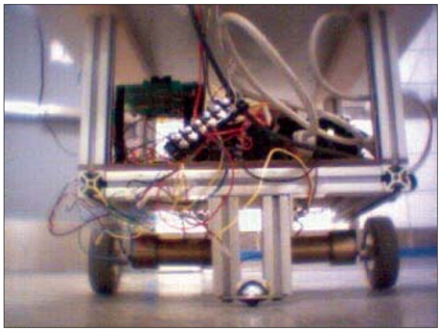


Figure 4. Rear castor wheel.

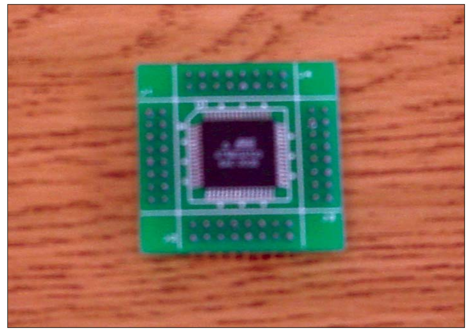


Figure 5. Atmel ATM103 MCU.

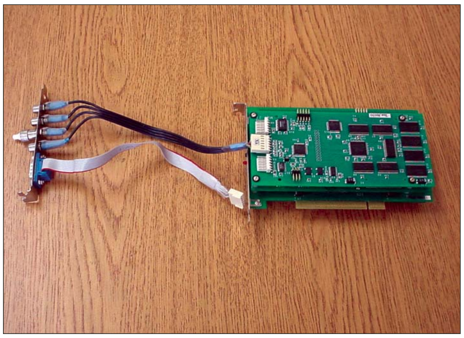


Figure 6. NM 6403 DSP board.

Figure 7. Panasonic 9V battery.

Figure 8. The inverter.

some results of the edge based door recognition algorithm. These images are converted to gray scale and then filtered to recover the edges.

By monitoring the relative percentage change in the edges, a door is recognized. When a drop in the relative percentage below a particular threshold (-0.03) is recognized, the robot is assumed to have encountered a door.

Figure 22 shows a plot of recognized doors. The program maintains an internal count for the doors encountered, if the door recognition algorithm fails. In addition, adequate measures have been incorporated so that when more than one image of the same door is captured, the robot does not treat them as two different doors.

Once a door is recognized, the control is passed to the Recognition Module in order to recognize the door. For more information on the edge detection algorithm, please visit the RISCBOT website listed at the end of the article.

Door ID Recognition

As the robot passes a door, it scans images for locating the door plate. The door images are acquired using the NTSC camera mounted on the side (see Figure 23). Once the door plate has been located, the numeral character is extracted. The extracted character is then scaled to a standard size and topological features of the character are calculated and compared with elements of a library (trained set of features).

A match of a library bit string

Figure 10. Ultrasonic sensors.

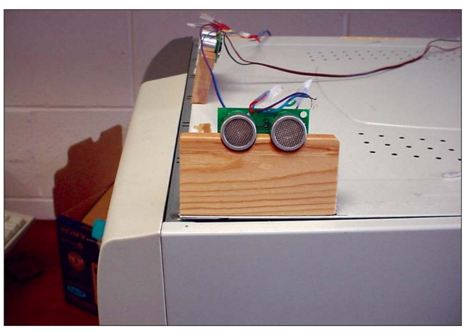


Figure 11. Onboard camera.

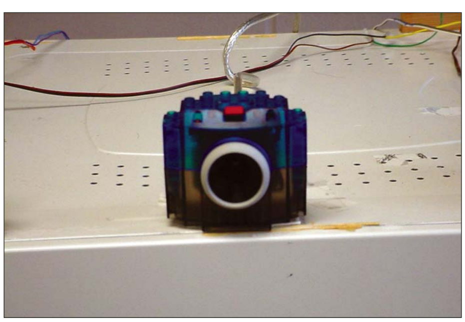
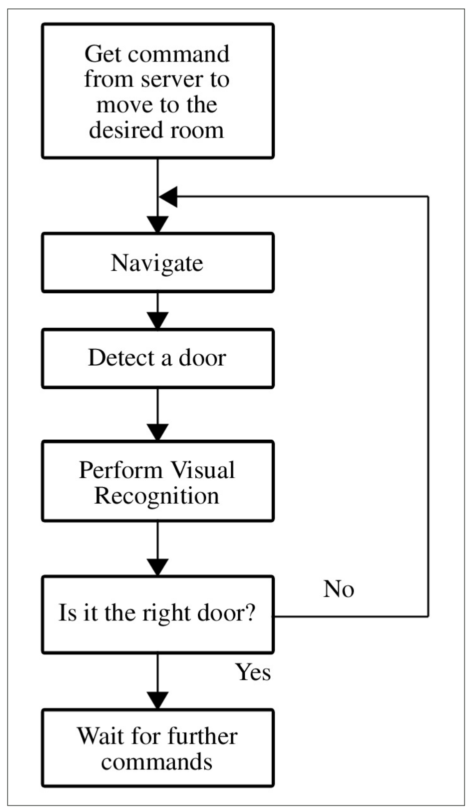


Figure 9. The navigation chart.



against an input string results in the corresponding digit class being assigned to the input digit. If this digit is the same as the desired door number, the robot stops or else continues to move towards the desired door.

The Web Interface

The web interface is an integral part of the mobile navigation and identification process. The mobile robot is connected to the Internet through an onboard WLAN 802.11b card which connects to the nearest wireless service providing an access point.

The robot can be controlled and viewed from the Internet, through the RISCBOT website. Updates on the web services and server availability information will be posted on the website. Users can also download videos and pictures of sample navigation and recognition tasks performed by the robot.

The RISCBOT web interface is simple, consisting of three windows: the control window, top view window, and the camera window. Figure 24 shows a view of the web interface while the robot is navigating.

The control window shows the instantaneous position of the RISCBOT. Once logged on, any remote user can command RISCBOT to move to a desired door by selecting one of the 11 listed doors on the right of the control window. When the user presses the MOVE button, the respective door number is sent to the RISCBOT server which then sends the command to the RISCBOT computer via the wireless link. The RISCBOT website is hosted on a Microsoft IIS Server supporting ASP for user interaction with the robot.

The main purpose of the other two windows are as follows: The top view window and the cam window provide visual feedback. The top view window shows the position of RISCBOT in the corridor. This view is the same as that on the command window. This video is provided by a network camera mounted on the ceiling of the corridor.

The cam view window provides a head-on real time video feedback from the robot as it moves through the corridor. The onboard camera continuously grabs image frames from the USB camera and wirelessly transmits them to the server. This feedback has been implemented using Microsoft Media Encoder.

Computational Time and Efficiency

The speed of the robot varies from 50 cm/s to 1 m/s, depending on the battery condition. The wall detection

Figures 12-16. Images captured by the robot.

Figures 17-21. Images after edge detection.

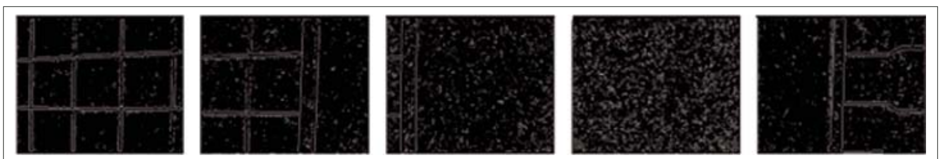


Figure 23. NTSC camera used for character recognition.

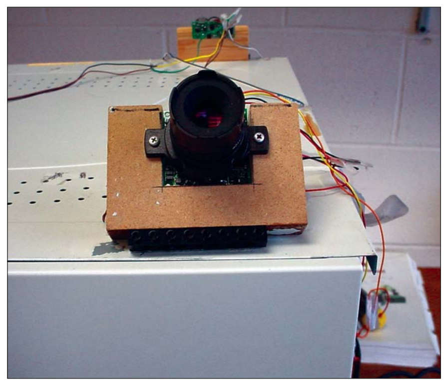


Figure 22. Plot showing the doors recognized.

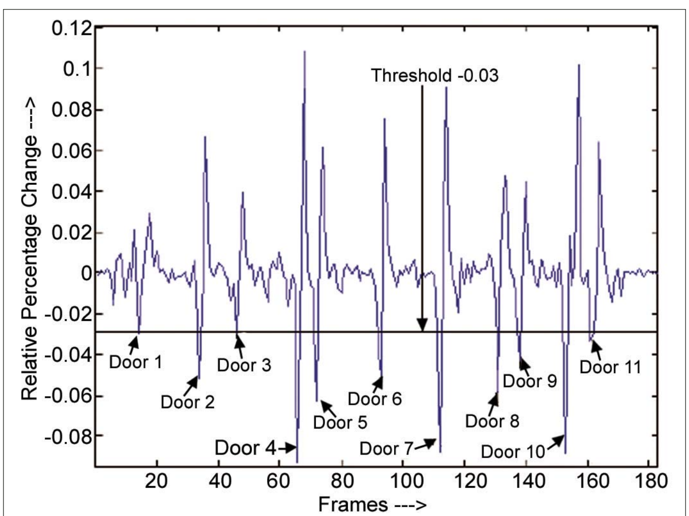


Figure 24. The RISCBOT website.

time using the ultrasonic sensors varies from 75 ms to 100 ms, depending on the distance of the robot from the wall. A time lag of 10 ms was employed between the firing of two successive ultrasonic sensors to avoid crosstalk.

Experimental results with the door ID recognition algorithm running in isolation under ideal conditions showed an efficiency of 99%. Experimental results with the character recognition algorithm running on the robot showed an efficiency of 80%. The failures (no decision on a particular door ID) were due to partial or no acquisition of the door plate, due to the proximity of the robot to the door.

However, while the robot catered to the requests of online users, an efficiency of 95% was recorded. An internal door count algorithm running in a MATLAB program overcame any failure of the character recognition algorithm. For more information and publications about RISCBOT, please visit **www.bridgeport.edu/sed/risc/html/proj/RISCBOT/index.htm**

Future Work

We are in the process of building RISCBOT II to collaborate with RISCBOT. RISCBOT II will have greater sensing and manipulative capabilities. We plan to incorporate the following features in RISCBOT II:

- Implementing data transfer algorithms for faster transfer of high priority data packet.
- Developing fast computing algorithms for the image-processing module.
- Exploring ways to enhance RISCBOT's cognitive and sensing capabilities.
- Implementing a DC-DC (ATX power supply) converter circuit that will increase the power conversion efficiency and thereby the operational time for the robot.
- Permitting the robot to recharge itself by plugging into wall outlets.
- Mounting a manipulator on the mobile platform for implementing mobile manipulation tasks.

Concluding Note

This article described the incipient developments in RISCBOT, a mobile robot platform developed at the University of Bridgeport. Researchers at the RISC lab are currently working on an arm to be fitted on the robot and further developing the vision and control algorithms that will enable the robot to plug itself to various power outlets. Readers are welcome to pose any queries or make suggestions to the authors. **SV**

About the Authors

This project idea has been conceived, guided, and supervised by Dr. Tarek Sobh. Dr. Tarek Sobh is the Dean of the School of Engineering at the University of Bridgeport and also heads the RISC lab. You can email him at sobh@bridgeport.edu

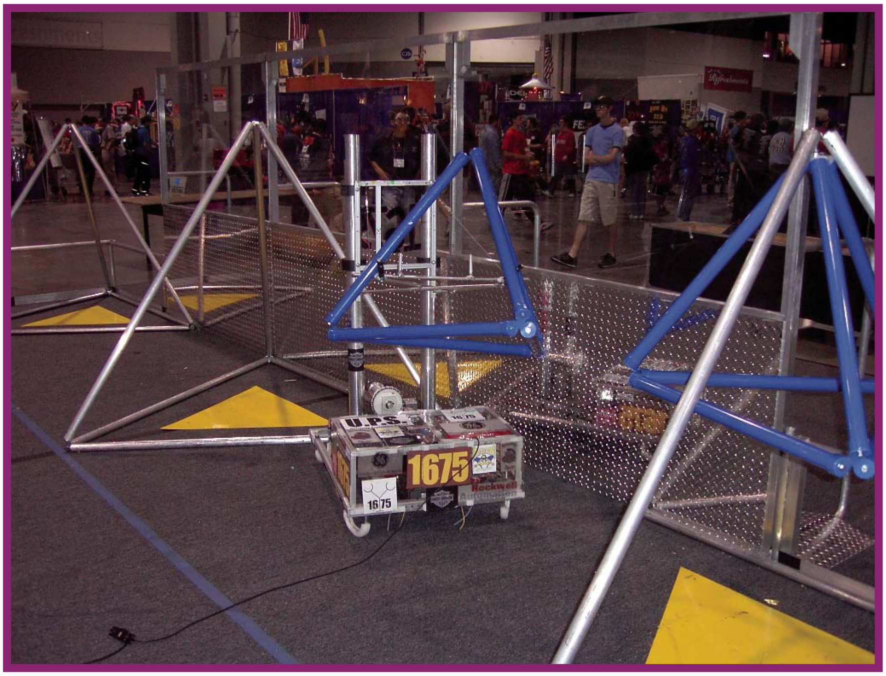
Sarosh Patel worked on the mechanical design and construction, web user interface of RISCBOT, wireless LAN interfacing, and image processing algorithms of the RISCBOT. He is pursuing his Master of Science in Electrical Engineering and Technology Management at the University of Bridgeport. You can email him at saroshp@bridgeport.edu

Rajeev Sanyal has worked on the embedded software, electronic hardware, mechanical design and construction, and image-processing software associated with RISCBOT. He recently graduated from the University of Bridgeport with a Master of Science in Electrical Engineering. You can email him at rajeevs@bridgeport.edu

Quick and Easy

AUTONOMOUS OPERATION *for* FIRST Robots

by Brian Cieslak



When our Team 1675 — The Ultimate Protection Squad — arrived at the FIRST Robotics Competition Midwest Regional last year, our robot was not programmed to operate during the 15-second autonomous period that starts each match. Being rookies, we had our hands full just getting the robot built and in working order. We didn't have time to add a lot of bells and whistles. **"Fifteen seconds seems like an eternity,"** team coach Kevin Kolodziej told me, as he sat and watched the robot sit idly on the playing field while other robots moved about scoring points.

LAST YEAR'S FIRST COMPETITION was a sophisticated tic-tac-toe game called Triple Play, which required teams or alliances of robots to stack geometrically shaped objects called "tetras" on top of goals (referred to as capping a goal) in order to score points. Additional points were scored if your alliance could "cap" rows and diagonals of goals. Each match started with 15 seconds of autonomous operation; in other words, no human input for control. After 15 seconds, drivers took over and controlled the robot's actions over an RF link by manipulating a set of joysticks.

Despite the lack of autonomous capabilities, Team 1675 was awarded the "Rookie All Star Team" award at the Midwest Regional. This award came with an automatic invitation to the National Competition, held in the Georgia Dome in Atlanta. In order to compete at this level, it was imperative that the robot be capable of scoring points during the autonomous period; otherwise, we would be forced to play catch-up during the whole match. To make the challenge even more difficult, we would not have access to the robot until we met again in Atlanta, in three weeks' time.

One day, while watching our drivers practice with a robot that we borrowed from another team, I noticed they were getting quite proficient at capping goals and knocking "hanging" tetras loose from the back row goals, both of which were techniques used for scoring during the autonomous mode of the contest. One of the kids on the team commented about our driver, "If we could only stick his brain in the robot, then he could drive from inside."

That was the inspiration for a program we dubbed AutoFlex, for flexible autonomous programming. By simply

FIGURE 1. This is a photo of me confirming the code.

adding a few files to our current program, we could record the control commands sent by the drivers as they manually executed the tasks required of the robot during an autonomous operation. With a quick recompile of the robot's control program, our robot could faithfully reproduce the training session when it was switched into "autonomous mode." This allowed us to quickly train our robot on the practice field when we were reunited in Atlanta.

Our robot was "autonomous-ready" by early afternoon on the day before the competition was to begin. The result was 13 autonomous points scored in seven matches, and with the help of our alliance partners, we scored back-row "Triple-Plays" in the first 15 seconds of each match. Team 1675 finished 24th in the Galileo division. Not bad for a bunch of beginners.

Adding AutoFlex to Existing Code

Like most rookie FIRST programmers, we modified the default code provided by the folks who make the robot controller at IFI (Innovation First, Inc.). All of our robot control functions are found in the Default_Routine function of the user_routines.c file.

Since our driver-controlled robot algorithms were complete and worked well, we simply had to add some routines to capture and play back the joystick commands.

Driver Control

The structure of the default code that is executed during driver control is shown in Figure 2A. The process_data from_master uP function, which is found in the user_routines.c file, processes joystick commands that are received over an RF link. The inputs are then processed by algorithms found in the Default Routine function. Results of the processed input data are applied to output hardware like motor drivers, relays, or LEDs.

In Figure 2B, you can see how we added a function to capture and record the input data as our drivers would send commands to the robot.

Example 1

Adding AutoFlex Capabilities

Add these files to the existing project :

AutoFlex.c
AutoFlex.h
Command_table.c

Autonomous Control

A special command signal received from the operator interface causes the robot to execute from the user_autonomous_code function instead of the process_data from_master µP function. The user_autonomous_code function is found in the user_routines_fast.c file. The structure of the autonomous code is similar to the driver control code. However, the input data is set to all neutral values so basically there are no driver commands getting to the robot. Without some form of code reassigning values to the command variables, the robot would just sit idly until the autonomous signal was no longer asserted.

In Figure 3, you can see where we added the autoflex_playback function that would read previously-recorded driver commands from memory and apply them to the command variables — just as if the drivers were sending them. The processed commands are applied to the same Default_Routine function that is called when drivers

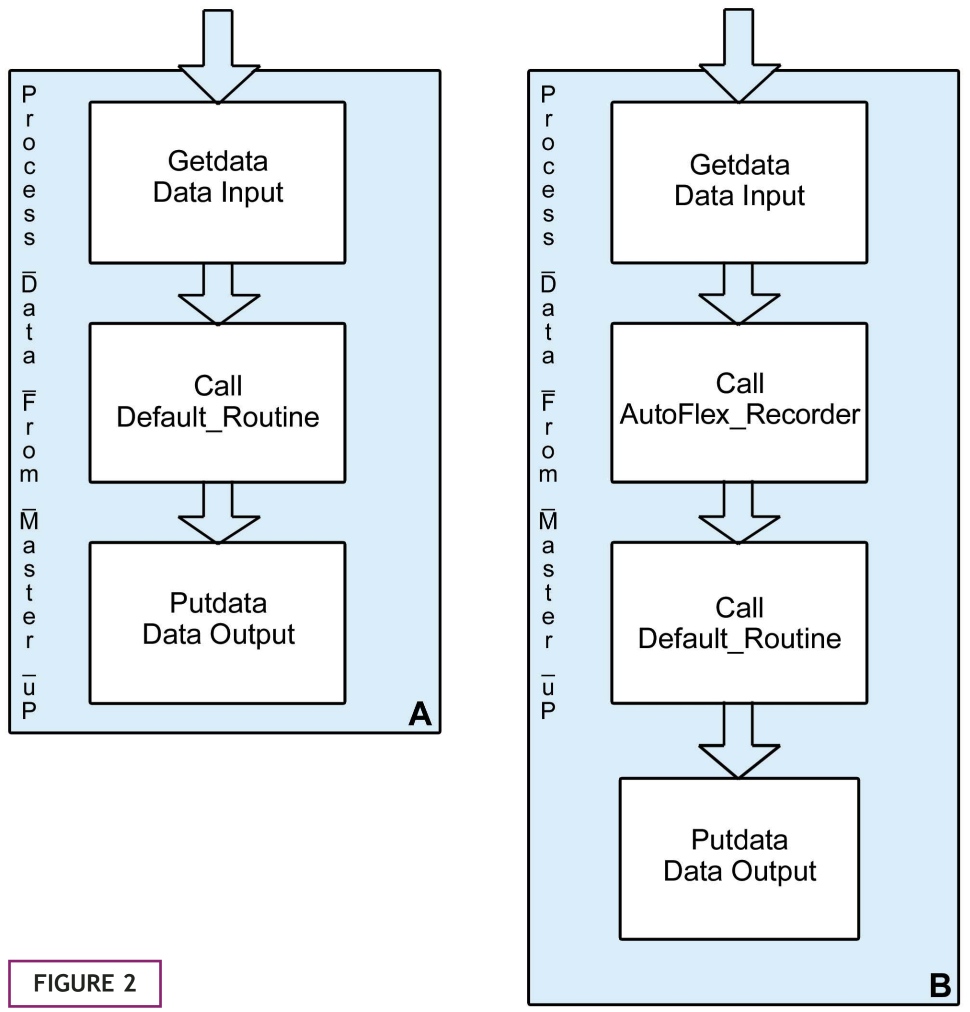


FIGURE 2

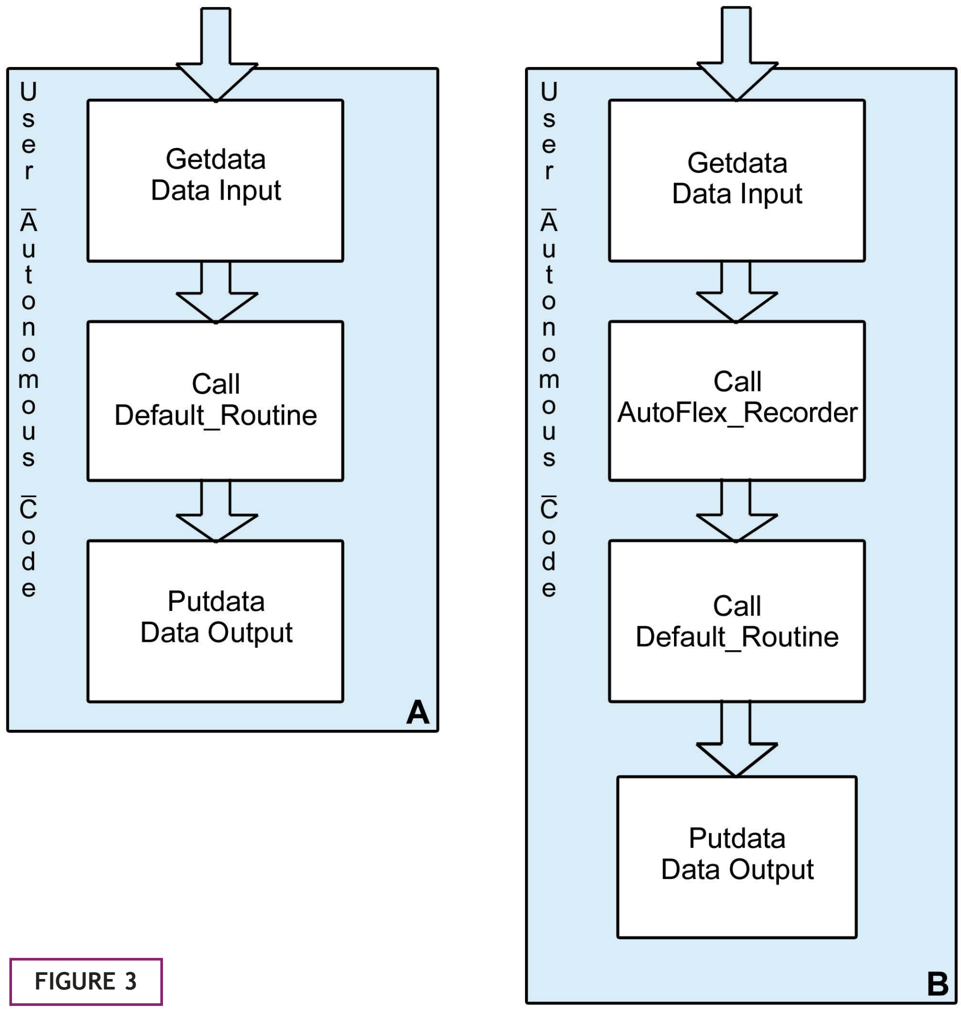


FIGURE 3

control the robot. The result is a reasonably accurate re-enactment of a recorded session.

Example 2

```
void Process_Data_From_Master_uP(void)
{
  static unsigned char i;

  Getdata(&rxdata);   /* Get fresh data from the master microprocessor. */

  /* Add your own code here. */
  autoflex_recorder();

  Default_Routine();  /* Optional. See below. */

  Generate_Pwms(pwm13,pwm14,pwm15,pwm16);

  Putdata(&txdata);           /* DO NOT CHANGE! */
}
```

Adding Files and Changing Your Code

1. *Open MPLAB and add three new files to the project* (see Example 1).

2. *Modify the user_routines.c file.* Add a call to the autoflex_recorder function in the Process_Data_From_Master_uP function. Also add a #include "autoflex.h" reference that includes the prototype for the autoflex_recorder function (see Example 2).

3. *Modify user_routines_fast.c file.* Add a call to the autoflex_playback function in the autonomous_user_code function. Also add a #include "autoflex.h" reference that includes the prototype for the autoflex_playback function (see Example 3).

The AutoFlex Recorder Function

The AutoFlex_Recorder function intercepts commands sent over the RF data channel by human drivers during the manual mode of operation. Every .01 second, the values of the joystick commands are read. The recorder outputs the data via the serial programming port in a format that builds a text file that can be compiled to create a permanent record of the session.

The text file, named command_table.c, is really a declaration of a multidimensional array, that when compiled is stored as a table of constants in program memory. Each line in the array is a snapshot of the joystick inputs taken at .01 second intervals. There are 150 rows in the array, providing 15 seconds of command data.

Our robot uses four joystick inputs for control, so each row of the array contains four elements (see Example 4).

Refer to the Command_table.c source listing. When the joystick is centered, or neutral, the command value sent is 127. Commands can range from 0 to 255 when the robot is being driven.

Training the Robot

Practice, practice, practice. That's what our human drivers did on the practice field in Atlanta until they could run through the autonomous tasks in their sleep. When we were confident we could go through the routine in one take, we were ready to train the robot.

Recording the text file requires that a laptop be connected to the

programming port of the IFI robot controller. A terminal program that can capture text is required. Windows Hyper Terminal program works well. Set the baud rate to 115,200 and flow control to none.

To start capturing data on the laptop, click on the "Transfer" tab and select the "Capture text" menu item. Assuming that our project is in the directory "C:\autoflexlite," enter the filename"c:\autoflexLite\command_table.c". It is important that this file is stored to your project directory. Make sure the human drivers are all set and that everything is ready to start and then click the "Start" button. The laptop is now armed and ready to start capturing data from the robot controller.

To start recording, click the trigger on the Port 1 Joystick. The recording session starts and the clock is running. You have 15 seconds to record your routine. You will see a line of data appear once every .01 second on the laptop display. Recording stops automatically after 15 seconds. At the end of the recording session, click on the "Transfer" tab and then the "Stop" menu item of the Hyperterminal program. All the data is saved to the file "C:\autoflexLite\command_table.c".

```
 void User_Autonomous_Code(void)
{
 /* Initialize all PWMs and Relays when entering Autonomous mode , or else it
   will be stuck with the last values mapped from the joysticks . Remember,
   even when Disabled it is reading inputs from the Operator Interface .
 */
  pwm03 = pwm04 = pwm05 = pwm06 = pwm07 = pwm08 = 127;
  pwm09 = pwm10 = pwm11 = pwm12 = pwm13 = pwm14 = pwm16 = 127;

//  pwm01 = pwm02 =  pwm05 = 127;  /* these will have to be selected to match our
robot */
//  relay1_fwd = relay1_rev = 0;

  relay2_fwd = relay2_rev = 0;
  relay3_fwd = relay3_rev = relay4_fwd = relay4_rev = 0;
  relay5_fwd = relay5_rev = relay6_fwd = relay6_rev = 0;
  relay7_fwd = relay7_rev = relay8_fwd = relay8_rev = 0;

 while (autonomous_mode)   /* DO NOT CHANGE! */
 {
  if (statusflag.NEW_SPI_DATA)      /* 26.2ms loop area */
  {
     Getdata(&rxdata);   /* DO NOT DELETE, or you will be stuck here forever! */

     /* Add your own autonomous code here . */
     autoflex_playback();

     Generate_Pwms(pwm13,pwm14,pwm15,pwm16);

     Putdata(&txdata);   /* DO NOT DELETE, or you will get no PWM outputs! */
  }
 }
}
```

Example 3

Transferring the Intelligence to the Robot

The command_table.c file has all the new command data in it. Rebuild the project using MPLAB.

Use the IFI_downloader to download the <filename>.hex file.

We gave each recorded session a unique name, like score_from_left.hex ,score_from_right.hex and score_from_center.hex. The name reflected our starting positions, which we download before each match.

The AutoFlex_ Playback Function

The autoflex_playback function is activated when the robot is placed in autonomous mode. A robot will run autonomously if:

1. The autonomous mode pin of the competition port on the operator interface is grounded. This occurs during the first 15 seconds of competition and is controlled by the contest directors. During development, the programmer can also ground this pin to initiate autonomous operation for testing and debugging.

Example 4

Operator Controls

P1_y	left wheel motor	controls	pwm01
P2_y	right wheel motor	controls	pwm02
P3_y	crane up / crane down	controls	pwm05
P4_x	crane left/crane right	controls	relay1

For Your Info

For the complete source listing, go to the *SERVO* website at **www.servomagazine.com**

2. The robot does not have an established communications link with the operator interface. This can happen if the operator interface is not powered-up, if there is no radio link, and/or the robot is not tethered to the operator interface.

For safety's sake, always have the operator interface powered on and tethered to the robot when developing autonomous code. Otherwise, the robot will start executing its autonomous routine when you turn it on. There's nothing more embarrassing (or dangerous) than turning on your robot and have it take off down the hall.

The autoflex_playback function will read one line of command values from the command_table every 100 ms and apply it to the associated hardware. If the autonomous mode is still being asserted after 15 seconds, all the commands will be go to the neutral value, basically deactivating the robot.

Before We Test

A valuable tool for developing autonomous functionality and robot training is a "dongle" or special connector to plug into the competition port of the operator interface. This will provide you with two switches. One is to place your robot into autonomous mode, just like at the start of a match. The second switch is a disable switch that you can hit if your robot runs amok. Plans for the dongle can be found at the IFI website:**www.ifirobotics.com/docs/competition-port-pinout-guide-reva.pdf**

Ready, Set, Go!

You've recorded an autonomous session, rebuilt the program, and downloaded it into your robot. Your operator interface is powered up with a dongle in place. All that's left to do is unleash your robot. Remember to put your robot in the same starting position it was in when you recorded your session.

Hit the autonomous switch on the dongle and hold on to your hat. If all goes well, your robot should re-enact the training session that you recorded. Congratulations! You can now impress potential alliance partners with your autonomous operation. **SV**

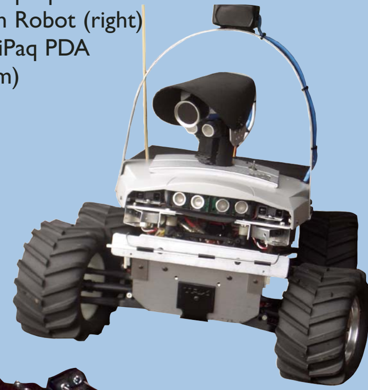
About the Author

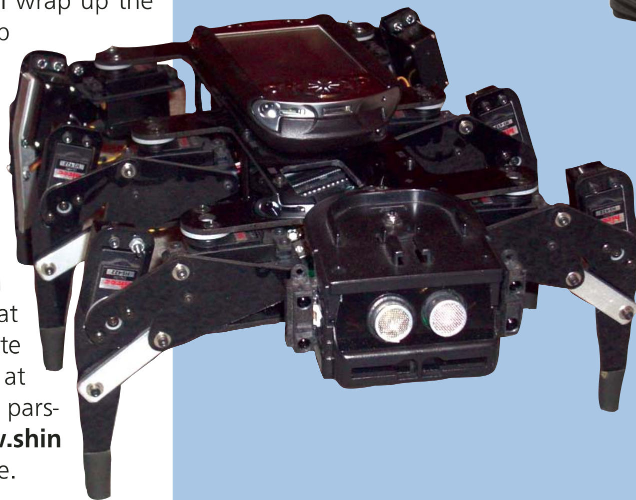
Brian Cieslak is a mentor for FIRST Team 1675, The Ultimate Protection Squad. He holds degrees in Physics and Computer Science and is a Senior Software Engineer with Rockwell Automation. His hobbies include Robotics and Amateur Radio (K9WIS).

[Part 3]

INTERMEDIATE ROBOTS

Building a Laptop- or PDA-Based Robot





MEET THE 'BOTS

- HelmBot — iPaq PDA Robot (left)
- Seeker — Laptop-based Robo-Magellan Robot (right)
- BugBot — iPaq PDA Robot (bottom)

In the first two parts of this series, I described the hardware design and core PC software for a laptop-based robot. In this final article of the series, I will finish up on the laptop software with a discussion of GPS and color tracking. Then, I will describe the PIC microcontroller software, and explain how the laptop and PIC communicate. Finally, I'll wrap up the series with some information on how to replace the laptop with a PocketPC-based PDA, such as an iPAQ.

Reading GPS

The laptop communicates with a GPS receiver via serial over USB, RS-232, or Bluetooth, using a protocol known as NEMA 0183. This standard was developed by the National Marine Electronics Association (NMEA). I had started writing an NMEA parser state machine when I discovered a great version written by Monte Variakojis of VisualGPS, LLC. Monte has an excellent white paper describing the parsing of NMEA at **www.visualgps.net** With his permission, I have integrated his parser code into the Seeker project and made it available at **www.shinsel.com/robots** See the sidebar on GPS parsing to learn more.

BY DAVE SHINSEL

After parsing the GPS information, the Seeker software function GPSDegreesToRWInches() converts the reported latitude and longitude to the robot map coordinates, in inches from a fixed origin. The internal robot map may be "anchored" to the real world by associating any single GPS coordinate with its corresponding X,Y robot map coordinate. This then allows all other data such as path waypoints (described last month), obstacles, etc. to be entered as either GPS coordinates or X,Y coordinates relative to the anchor point.

The robot can then use the GPS receiver to track its location and navigate to any location on the map. Note, however, that GPS is generally not accurate enough for small robots. The position error can be 20 feet or more, so the GPS is usually only used to augment the compass and odometer in tracking the robot position. In addition, most GPS receivers do not work well (if at all) indoors, so GPS is mostly usable for outdoor robots.

Object Color Tracking

For the SRS Robo Magellan contest, the robot must be able to find and touch an orange traffic cone that has been placed at the end of the course. In addition, bonus time is awarded for touching other cones scattered along the way. The most common way to do this is with "color blob" spotting — find a blob with lots of orange, and it is probably a cone!

I considered using the popular CMU Cam for this, but was not really satisfied with the performance for spotting a cone outdoors at long range. In addition, I really wanted to do some work on the laptop with vision, with the idea that at some point I could add advanced vision processing, such as object avoidance and path navigation.

My first laptop robot — Mr. Roboto.

Mr. Roboto reading *Nuts & Volts*. Mr. Roboto roaming.

I had looked at several vision libraries when I discovered Robin Hewitt's Mavis project. This turned out to be a good fit for what I needed, but Robin had not yet added color blob tracking. Well, I went ahead and integrated (okay, hacked) Robin's library into my robot code, and then added a new ColorBlob object to the ObjRec (Object Recognition) class.

To implement the ColorBlob object, I needed a way to reliably detect a color, and reject all others. I had heard that the RGB (red, green, blue) values that most cameras provide are very difficult to use for color spotting, as any change to brightness causes all three values to change, and not uniformly! I had also heard that working in "YUV" color space works a lot better, but did not know how to go about converting from RGB to YUV.

Fortunately, a friend of mine, Matt Curfman, pointed me to the "FourCC" website at **www.fourcc.org** This website has a ton of information about common video formats and color space conversion. I was able to find a simple conversion formula to get to the pure red and blue color components, with all luminance (brightness) removed. The code boils down to the following, with Y representing luminance:

```
Y = 0.299*R + 0.587*G + 0.114*B;
// Clamp Y at valid values
if( Y < 16 ) Y = 16;
if( Y > 235 ) Y = 235;
Cr = 0.713*(R - Y);
if( Cr < 0 ) Cr = 0;
if( Cr > 255 ) Cr = 255;
Cb = 0.564*(B - Y);
if( Cb < 0 ) Cb = 0;
if( Cb > 255 ) Cb = 255;
```

Color purists may argue about the correctness of this. In fact, there are several conflicting versions posted at FourCC, but this works pretty well for this application. Note that the result is just two values: Cr (Component Red) and Cb (Component Blue). The green component can be calculated from the red and blue, so is not needed.

The actual color tracking code is quite simple. The camera is pointed at a sample of the color you wish to track, and the Cr and Cb values are stored as the target color. In operation, each video frame is quickly converted from RGB to Cr/Cb values, and then the frame is searched for values where both the Cr and Cb are within a pre-defined threshold of the target color. If enough pixels are found that "match" the target color, an object is considered found. The average X and Y of all the matching pixels is computed during the search, and returned as the center of the color blob.

Once the color blob X,Y coordinate is known, the location is sent from Mavis to the Robot Camera Control Module. The distance of X,Y from the center of the frame is used to calculate the amount of servo travel needed to center the color blob in the video frame.

A BRIEF INTRODUCTION TO GPS DATA FORMAT

Here are a few "NMEA Sentences" (as they are called) from a typical GPS receiver:

$GPGSA,A,1,31,03,25,29,,,,,,,,,11.9,6.1,10.2*30
$GPGSV,3,1,10,29,70,130,37,28,53,328,,15,53,026,,21,47,101,*73
$GPGSV,3,2,10,08,28,121,,25,27,014,39,14,25,253,,31,23,292,41*7D
$GPGSV,3,3,10,09,09,143,,03,07,334,36*7B
$GPRMC,054105.998,V,4250.5461,S,14718.4860,E,0.26,185.02,211200,,*0A

You got all that? Well, it's actually not so bad. The last sentence shown is one of the most common sentences: RMC, or "Recommended Minimum Navigation Information." RMC is formatted as follows. Each comma represents one "field" as follows:

Field:	1	2	3	4	5	6	7	8	9	10	11	12
$GPRMC	054105.998	V	4250.5461	S	14718.4860	E	0.26	185.02	211200			*0A

For RMC, the fields are defined as follows:

1. Time (Universal Time Code)
2. Status (V = Navigation receiver warning)
3. Latitude
4. N or S
5. Longitude
6. E or W
7. Speed over ground, knots
8. Track made good, degrees true
9. Date, ddmmyy
10. Magnetic Variation, degrees
11. E or W
(* instead of comma)
12. Checksum

Generally, the robot mostly cares about the latitude, N/S, and longitude, E/W. This can be used to map the robot location anywhere on the planet, but there are many other NMEA Sentences that provide information such as direction quality of fix and number of satellites tracked.

For more information about NMEA Sentences, see **www.gpsinformation.org/dale/nmea.htm**

*Other names and brands may be claimed as the property of others.

This information is sent to the camera tilt and pan servos, causing the camera to track the object nicely. The Robot Navigation Module uses the camera pan servo position to drive the robot to the object.

There is plenty of room for improvement to the current implementation. First on my list is to add a region search to make sure enough of the target pixels are clumped together to assure a valid object. Second is to check the object outline shape, to make sure that the robot is not tracking somebody's jacket! However, despite its limitations, the current implementation works surprisingly well. Take a look at Figure 1 and notice the red cross hairs on the cone. The cross hairs indicate the center of the tracked color blob.

If you are interested in learning more about vision processing, I highly recommend that you read the "Getting Started with Vision" series of articles by Robin Hewitt, starting with the July 2005 issue of *SERVO Magazine*.

PIC Microcontroller Software

Remember from the last article that the PIC acts as a "slave" to the laptop software. All higher-level decisions are made on the laptop, while hardware control, sensor input, and timing-critical tasks are performed by the PIC. All control actions are in response to a command from the laptop.

PIC Main Loop

The PIC software is designed around two loops that run continuously. You are probably already familiar with the concept of a "Main" loop. Most small robot controllers follow this basic design:

- Initialize software variables.
- Initialize hardware.
- Start of Loop
  - Read sensors.
  - Decide what to do.
  - Issue motor commands.
- Go to Start of Loop.

For a laptop-based robot, the PIC Main Loop is modified to handle serial communication with the laptop:

- Initialize software variables.
- Initialize hardware.
- Start of Loop
  - Check for Serial Command received from the Laptop. If a command is ready, process the command. If the command is Get_Status, send the current status to laptop.
  - Every 20 ms, read A2D sensors (IR rangers, battery level, etc.), and keep a running average.
  - Every 40 ms, read Compass and keep a running average.
- Go to Start of Loop.

Note that the decision of "what to do" is now handled by the laptop. Also, note that sensor readings only take place as often as each sensor is able to update. Even so, most sensors may be read many times faster than the laptop requests the sensor data. The program takes advantage of this to average the sensor readings on the PIC to improve sensor accuracy.

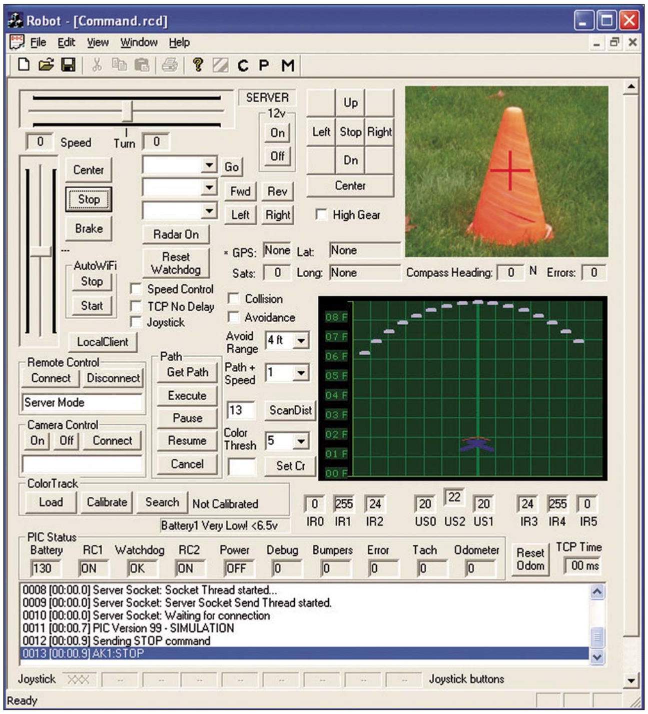
PIC Timer Loop

For tasks that require precise timing — such as servo control — a hardware timer (Timer0) is used to form a loop made up of 1 ms "ticks." At each timer interval, the Main Loop is interrupted, a single "tick" in the Timer Loop is executed, and then control is returned to the Main Loop.

The Timer Loop syncs around a 20 ms cycle (convenient for servo control) as follows:

- Base Code (Executed each time):
  - Increment/decrement timers.
  - Check wheel odometer for black/white transition.
  - Check ultrasonic capture register for echo received.
- T0: 400 µS duration
  - Copy servo values to counters.
  - Transition high for all enabled servos.
  - Start ultrasonic sensor pulse.
- T1: 3 ms duration
  - Go low when appropriate for each servo. This allows for full servo travel, with fine granularity.
  - During this period, execute Base Code every 1 ms to keep timings correct.
- T2: 600 µS duration
  - Pad to get back to an even 1 ms cycle.
- T3: 1 ms
  - Check bumper switches.
- T4: 1 ms
  - Check "dead man" RC control.
- T5: 1 ms
  - Every 200 ms, calculate tachometer and odometer.
  - Calculate speed control feedback and adjust motor speed.

FIGURE 1. Seeker Command GUI.



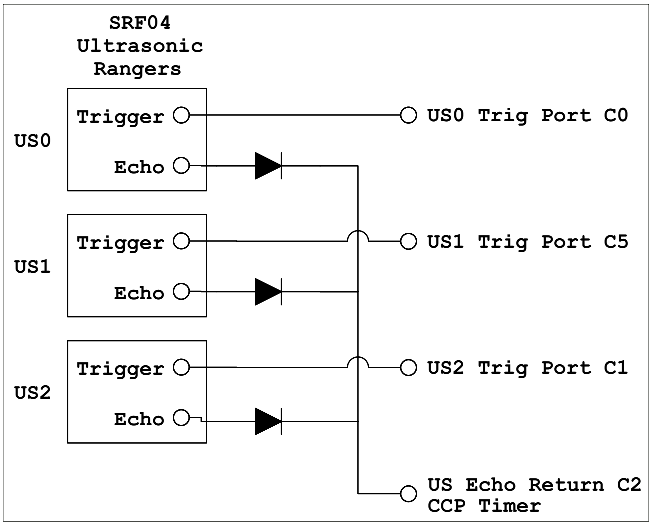


FIGURE 2. Ultrasonic Interface.

- T6: 1 ms
  - Handle slow servo movements (automatically increments servo position).
- T7: 1 ms
  - Handle motor "brake" control.
- T8, T9: 1 ms each
  - Not used.
- T10: 1 ms:
  - Set flag to allow reading of compass and A2D sensors in main control loop. This operation takes several ms so a good time to do this is while the Timer Loop is not busy.
- T11-T18: 1 ms each
  - Not used.

Note that there are only 19 "ticks," due to an extra long T1 time, but the total adds to 20 ms.

Ultrasonic Range Timing

The SRF04 is used for all ultrasonic ranging. Since sound travels at approximately 1.125 feet per millisecond, this module requires a fast timer to get accurate measurements. The PIC has a pair of CCP (Capture/Compare/PWM) registers that can be associated with one of the timers to measure a pulse width (among other things). I use one of these CCP registers for the ultrasonic sensors as follows:

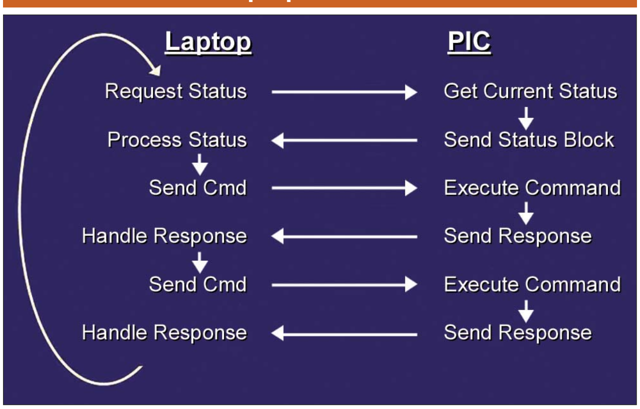
1. Raise the SRF04 trigger line high for at least 10 µS. This starts the Ultrasonic Burst.

2. Wait for the echo line to go high, indicating the Burst is done.

3. Start the CCP counter to time the echo response time.

4. Once each ms (in the PIC Timer Loop), check to see if the CCP register is set to non-zero, indicating that an echo was received, and the time of flight is stored in the CCP register. If so, copy the range value to the Status block to be sent to the laptop.

FIGURE 3. Laptop/PIC Communication.



The conversion from timer count to distance (in inches) is done on the laptop, along with all other conversions, such as IR range conversion.

The preceding method works great for one sensor. But how do you get multiple SRF04s to share a single CCP?

The solution turned out to be very easy. As you might recall from Part 1 of this series, each SRF04 has a separate Trigger line that is toggled to start a reading, but the Echo Return lines from all the SRF04s are "wired OR'd" together, using a diode on each sensor (see Figure 2). The software reads each SRF04 in "round robin" sequence by toggling a sensor's Trigger line, waiting for a response on the shared Echo Return line, and then proceeding to the next sensor. It is important during this process that at least 36 ms elapse between each sensor trigger event. (I give it 50 ms just to be sure.) Otherwise, when the next sensor is read, it might inadvertently receive an echo from the previous sensor's ping.

Serial Communication

Refer to Figure 3. The Laptop and PIC communicate via RS-232 Serial. Communication between the Laptop and PIC is always initiated by the Laptop. The Laptop begins by sending the GET_STATUS Command (it does this every 100 ms, or 10 times per second). The PIC responds to this command by building a PIC_STATUS block and sending it back to the laptop.

As explained in last month's article, on the laptop the PIC_STATUS is sent to the Behavioral Modules which, in turn, issue a series of commands. For each command, the PIC may optionally send a response. This response is mostly used for debugging that the command was received and processed correctly.

A lot of data is being sent back and forth between the laptop and PIC. When you start pushing the limits of serial bandwidth as this design does, there are some issues that arise:

Issue 1: The laptop will buffer up commands, so one command can run right into another command. If a byte is dropped, the PIC won't know

where the start of each command is, and will misread the next command. This can be disastrous if the command was supposed to be stop, but was read as "go real fast!" The way I solved this issue was to provide sync characters and termination characters in the command.

The resulting command is structured as follows. (I had planned to replace the Termination Character with a checksum, but never got around to it, and have not found it necessary so far.)

Byte Command
1. Sync0 (I use 0xE5)
2. Sync1 (I use 0x5F)
3. Command Byte
4. Parameter 1
5. Parameter 2
6. Parameter 3
7. Parameter 4
8. Termination Character (I use 0xC4)

If one of the framing characters (Sync0, Sync1, or Term) are not received, the command is discarded, and the serial bit stream is scanned until the next valid Sync0 is found.

Issue 2: The PIC has a very small receive buffer (typically three bytes), which will overflow if not handled quickly enough. This limitation is overcome by using the Serial Read Data Interrupt, INT_RDA. As soon as incoming serial bytes are received, they are parsed and placed into the Incoming Command structure by the read data Interrupt Service Routine (ISR), CheckForSerialData().

However, there is one "gotcha." The PIC does not handle "nested interrupts." For example, if the PIC Timer Loop is handling the 1 ms interrupt, and a serial byte comes in, the INT_RDA will never trigger. Therefore, I call the CheckForSerialData() function explicitly at the end of each Timer0 interrupt, assuring the receive buffer is checked at least once every millisecond.

Issue 3: No matter what you do, sometimes commands are dropped. My robot used to get stuck at times because the PIC never got the next move command. To resolve this, I tried a number of command retry schemes, (some quite convoluted) and finally found one that works quite well.

The key was in realizing that commands to the PIC have a limited life. A command to turn 10 degrees might be replaced half a second later with a command to turn 15 degrees, so there's no sense retrying the first command if it has been dropped. On the other hand, an urgent Stop command really does need to get through!

The solution to this is that each Status returned from the PIC contains the current state for critical subsystems, such as motor speed and turn. The laptop keeps track of the current desired state, and compares it to the one reported by the PIC. If they do

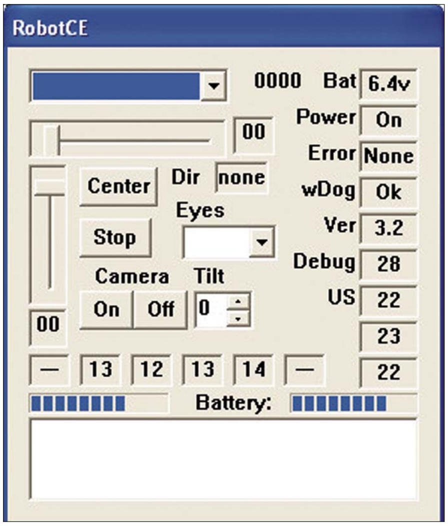


FIGURE 4. HelmetBot iPAQ GUI.

not agree, the laptop re-issues the command. This happens up to 10 times per second, until the PIC confirms that it is in the state that the laptop expects.

Using a PocketPC PDA

There are two main types of Personal Data Assistants (PDAs) typically used for robots: Palm PDAs and PocketPC PDAs. Since I was already using Microsoft Visual C++ for my laptop software, I chose to use iPAQ PocketPC PDAs for my smaller robots, HelmetBot and BugBot. PocketPC PDAs run Microsoft WindowsCE (usually referred to as "WinCE"). The development environment I use for WinCE is Microsoft eMbedded Visual C++.

One really cool feature of this environment (besides the fact that it is just like C++ for Windows) is the remote debugger. If your iPAQ has a wireless card, you can compile a new program, download it to the iPAQ, and step through the code as it executes — even if the robot itself is sitting in the next room!

To get started developing on a PocketPC, I suggest the following: First, buy a used PocketPC on eBay. I have seen the iPaq H5550, which has wireless built in, sell for under $200. If you don't need wireless, older iPAQs, such as the 3765 may sell for under $100, but to add wireless to one of these, you will need a PCMCIA or Compact Flash 802.11 Wireless card, and an iPaq Jacket to plug it into. Make sure the iPaq has a good battery, as it is usually the first thing to go bad (but if it does, you can buy replacements on the Internet).

Next, I highly recommend buying the book *WindowsCE 3.0 Application Programming* by Grattan and Brain (ISBN: 0130255920). This excellent book has a ton of information you need to know for programming WinCE, and Microsoft eMbedded Visual C++ is included on CD-ROM!

You will find that 90% of your robot code is sharable between a laptop robot and PocketPC robot. The main difference is the Graphical User Interface (GUI). See Figure 4 for HelmetBot's GUI. To go from a laptop robot based on Win32 to a WinCE based robot, I usually split the code into a Win32 Client project, which stays on the laptop or PC, and a WinCE Server project, which is used for robot control.

For the robot WinCE Server program, create a new WinCE project, design the GUI, and then start copying the robot engine program files into the new project. If you plan to implement common code for both WinCE and Win32, it's best to get the code working on WinCE first, since some functions available for WinCE are subsets of their Win32 cousins.

For example, if you look at my code, you will notice most strings are surrounded by the "_T()" macro. This macro forces strings to be in Unicode, which is required for all WinCE strings. If you forget this, the code will compile for Win32 just fine, but will not compile for WinCE.

Programming Languages

As mentioned previously in this series, I don't currently use Microsoft .NET programming for either the laptop or WinCE robots. However, that does not mean you can't! I think most of the source code I have posted should port fairly easily to the .Net environment, but I just have not gotten around to it. Also, I know people that are using Java, Visual Basic, etc. Pick a language that works for you!

On the PIC Microcontroller, I use the CCS C compiler (available at **www.ccsinfo.com/pic**). I chose this because I like using C, and it's a fairly inexpensive compiler. The command line compiler works fine, and integrates easily into Microsoft Visual Studio. There are also various PIC Basic and other compilers available, many of them cheap or even free.

Conclusion

The subject of laptop robotics could easily fill a book. In this article, I provided a brief overview of several subjects in which I have seen the most interest. I hope that there is enough information provided to get you started building your own laptop- or PDA-based robot.

As stated previously, both a demo of the Seeker application and the hardware schematic are available on the *SERVO* website (**www.servomagazine.com**). Further, all the source code mentioned is available at my website at **www.shinsel.com/robots** Now, go build that 'bot! **SV**

ABOUT THE AUTHOR

Dave Shinsel built his first robot in 1980 (yes, before the IBM PC was released) using a 1 MHz 6502 processor and 9K of RAM. He has been a hardware and software engineer for a number of companies including Hughes Aircraft, Epson Printers, Mentor Graphics, and for the last 12 years, Intel Corporation. At Intel, Dave manages a software engineering team for the Consumer Electronics Group in Portland, OR. His degrees in Electrical Engineering and Computer Science are from Cal State Long Beach, CA.

by Dave Calkins

2005 ROBOT SOCCER Championships

Part 2 — FIRA Singapore

Last month, I wrote about RoboCup — the Japan-based robot soccer organization — most famous for Aibo soccer. This month, we'll be examining the Korea-based FIRA — the Federation International Robot-soccer Association. FIRA is a play on the word FIFA — the Federation Internationale de Football Association — which is the governing body of [human] soccer. Both FIRA and RoboCup would like to eventually create autonomous humanoid robots that can beat the FIFA world champions.

FIRA History

In 1995, professor Jong-Hwan Kim of the Korea Advanced Institute of Science and Technology (KAIST) wanted to push robotics development beyond its then-snail's pace. Knowing that competitions are the best way to motivate people, he started FIRA, hoping to attract researchers from around the world to build better, smarter, faster robots.

In 1996, the first event was held in Daejeon, Korea, and has continued to grow since then. Over the last 10 years, the international championships have jumped from continent to continent, hosted by major universities in France, Brazil, Australia, China, Korea, Austria, and Singapore. The University of Dortmund will hold the 2006 FIRA World Cup at the same time and place as the FIFA World Cup in Dortmund, Germany. The 2007 World Cup will finally make it to the United States, with San Francisco State University hosting.

The World Cup events also have a serious side, with peer-reviewed research papers covering everything from vision systems to bipedal walking algorithms.

FIRA's objectives are not solely to build cool robots. They want to:

- Take the spirit of science and technology to the young generation and laymen.

- Promote the development of autonomous multi-agent robotic systems that can cooperate with each other.

- Bring together skilled researchers and students from many different backgrounds into the new and growing interdisciplinary field of robotics.

- Organize the FIRA Robot World Cup and Congress every year.

- Work together to establish the FIRA Robot World Cup as a Science and Technology World Cup.

Varying Events

Much like RoboCup, FIRA does not have one single competition. They currently sponsor six major categories with 12 total competitions.

MiroSot (Micro Robot Soccer Tournament)

This is the big event. Robots are 7.5

> “Knowing that competitions are the best way to motivate people, [Jong-Hwan Kim] started FIRA, hoping to attract researchers from around the world to build better, smarter, faster robots.”


Photo 1


Photo 4

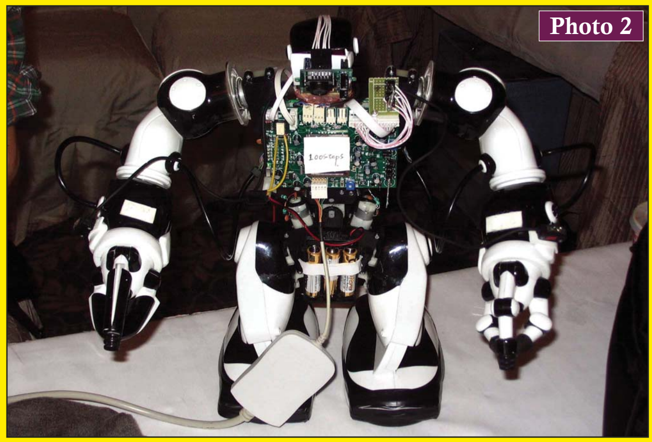


Photo 2

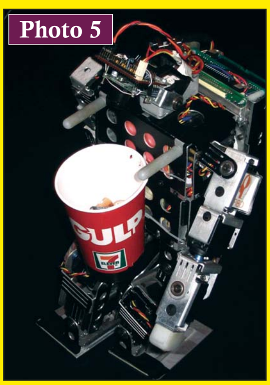


Photo 5

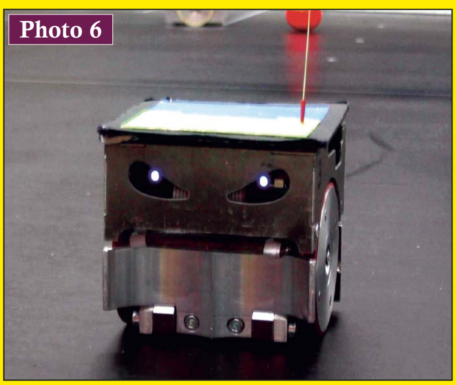


Photo 6


Photo 3

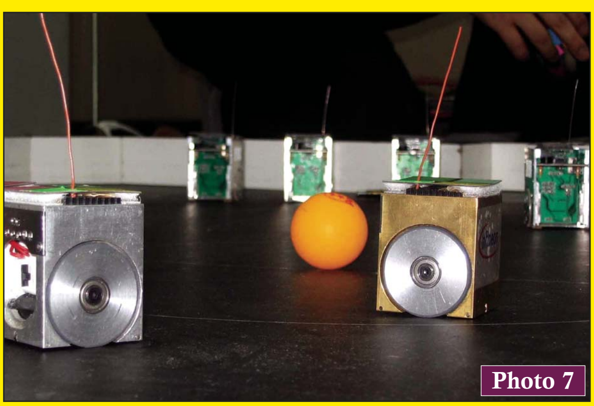


Photo 7

PHOTO 1. Team Singapore celebrates their wins.

PHOTO 2. Clearly, this bot would win *SERVO*’s Hack-A-Sapien contest ...

PHOTO 3. Getting ready for the kick-off in MiroSot 11:11.

PHOTO 4. The HuroSot dash.

PHOTO 5. A Canadian Robo-One with a PDA brain. Drink, anyone?

PHOTO 6. A snarling Team Germany MiroSot robot could scare Arnold himself!

PHOTO 7. Look at those NanoSot robots go!

cm on an edge, with three competitions: three-on-three, five-on-five, and eleven-on-eleven (standard human soccer games have 11 members per side). Bots are about the size of an orange (or a mini-sumo robot), but they are *fast*. In trying to video tape the event, I was forced to use a wide-angle lens because the robots move with lighting speed across the play-field. The ball is an orange regulation golf ball, and it shoots across the field of play as fast as if you were using a golf club.

The 11:11 MiroSot game is probably the most exciting robot soccer competition of any of the leagues. The robots are fast and plentiful (22 robots vs. the typical eight), the field is huge, the scores are high (unlike human soccer), and the field walled — so soccer play is much like billiards with angled shots deflecting around the field.

MiroSot robots are also probably the most complex bots in soccer. The actual robots are marked on their top with color patches to distinguish them from each other and the opposing team. Cameras installed above the field of play track the location of individual robots on both teams, the edges of the field and goals, and the ball. This video signal is then carried to each team's controlling PC, which interprets the location and vectors of both bots and ball, and then autonomously sends a WiFi signal to their own robots, telling them where to go and how fast. Programs must not only control the robots, but also do a good job of strategizing plays against the other team — including not bunching robots around the ball and anticipating ball travel.

NanoSot (Nano Robot Soccer Tournament)

NanoSot is almost identical to MiroSot, except robots are about 1/4 the size of MiroSot bots, and the ball is a ping-pong ball rather than a golf ball. The robots are so light that they have a bad tendency to flip over. Unlike MiroSot, NanoSot only has a five-on-five tournament, and the field is much smaller.

KheperaSot (Khepera Robo Soccer Tournament)

This is a one-on-one soccer game played with a tennis ball. Robots are usually enclosed in a soda can, and the robots are autonomous — looking for each other with on-board cameras and trying to push the ball into a goal. Since you only need to build one robot, it a great entry level point for people trying to start in robot soccer. A few companies offer kits on the web, so if you're looking to get involved in robot soccer, KheperaSot is a good starting point.

RoboSot (Robot Soccer Tournament)

RoboSot is very similar to KheperaSot, except that the robots and field of play are much bigger, and it offers both one-on-one and three-on-three variants. While the robots are almost four times the size of the soda-can bots, they still use a tennis ball. Anyone who's built a 3 kg sumo robot should seriously consider converting their sumo to be able to also play RoboSot (the bot specs are basically the same.) Obviously, you'll have to reprogram the robot to tell the difference between opposing goal and own-goal, and know which robots are on your team, but it would be a great exercise for the more advanced bot builders looking to play with color CMU cams.

FIRA 2005 CHAMPIONS

MiroSot 5:5
1. Socrates — Singapore
2. ICRO — Korea
3. AUSTRO — Austria

MiroSot 11:11
1. Socrates — Singapore
2. SPARIC — Singapore
3. ICRO — Korea

HuroSot
1. Manus — Singapore
2. Team RO-PE — Singapore
3. SCUT 100Steps — China

NanoSot
1. AUSTRO — Austria
2. ICRO — Korea

RoboSot
1. HIT — China
2. TKU — Taiwan
3. WIT — China

KheperaSot
1. Jumbo — Australia
2. Kheperoo — Australia
3. SCT Scooters — Germany

SimuroSot 5:5
1. CUG — China
2. WillWing — China
3. A.I.R. — Korea

SimuroSot 11:11
1. WIT — China)
2. CUG — China
3. NWPU — China

HuroSot (Humanoid Robot Soccer Tournament)

Unlike the other events, HuroSot is not actually a full soccer game. It includes four challenges:

- Robot forward/backward dash, which tests robots abilities to run without falling over;
- Penalty kick, which is the closest to actual soccer, but without all that annoying running up and down the field;
- Obstacle run is a timed event that tests bots ability to avoid objects in their path;
- The lift and carry is the iron-man competition to see how strong a bot really is.

SimuroSot (Simulation Robot Soccer Tournament)

Essentially, the same as RoboCup's Simulation League. SimuroSot does not actually involve physical robots, but software agents. A host server creates the environment (playground and robots — usually 3D videogame-esque human players) while remote player computers plug in with strategies and team AI. The goal with SimuroSot is not to develop better robots, but better strategies and AI that will eventually be built into the android team that will play against the actual humans. SimuroSot has both five-on-five and eleven-on-eleven games, and the simulation platform can be downloaded from the FIRA website.

2006 FIRA Soccer Events

If you're looking to either compete in FIRA, RoboCup, or to just watch a match, there are lots of events coming up. The European Championship will be held March 2nd to March 5th, 2006 in Vienna, Austria. The 2006 FIRA World Cup is going to be held in Dortmund, Germany from June 30th to July 3rd, and in the US, ROBOlympics will — once again — host multiple robot soccer events June 16-18 in San Francisco, CA, along with the *SERVO*-sponsored 2006 Tetsujin event, robot combat, and many other events. Next month — Roboexotica — the international robot bartending competition. SV

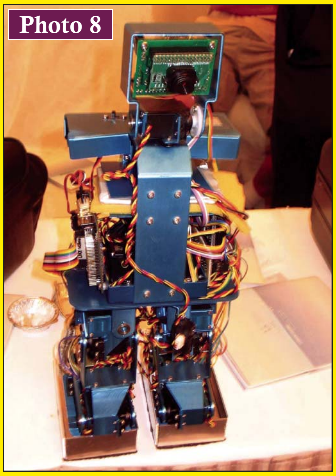


Photo 8

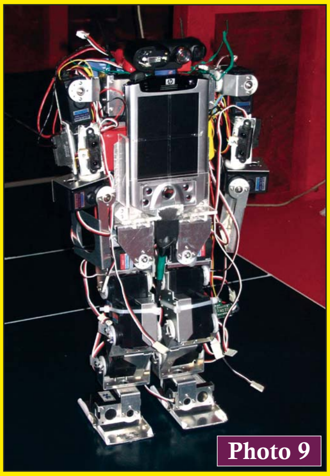


Photo 9

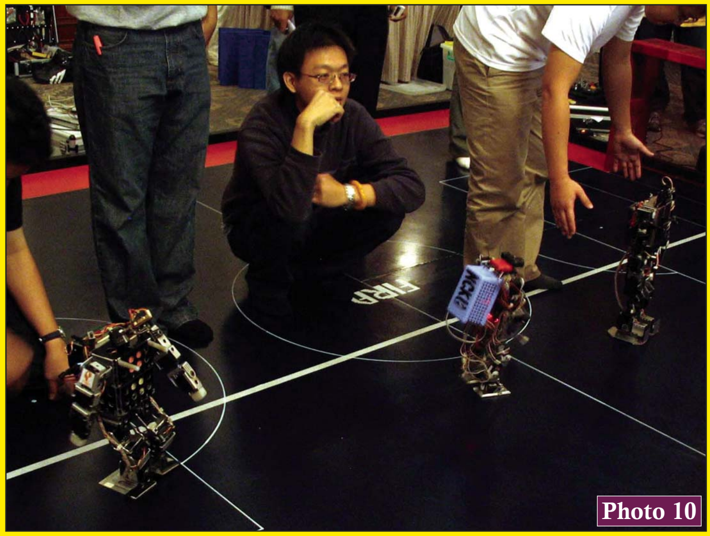


Photo 10

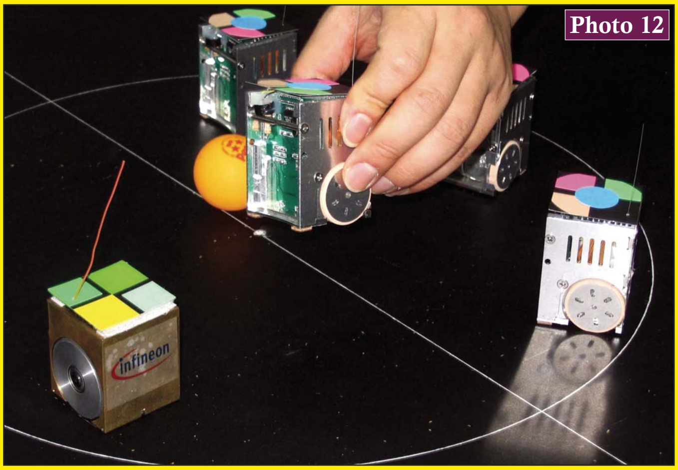


Photo 12

Photo 11

Photo 13

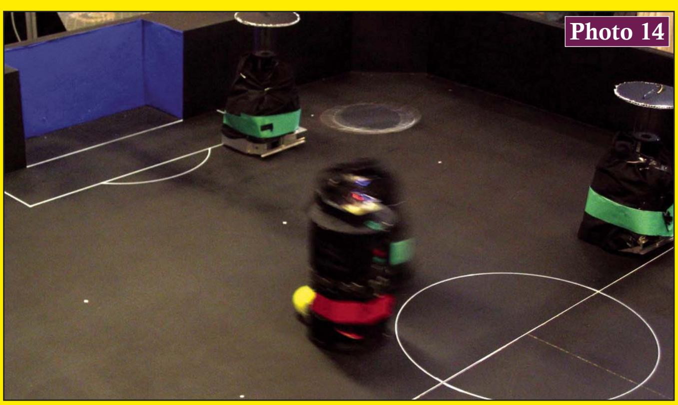


Photo 14

Photo 15

PHOTO 8. Homebrew humanoid bot from Team Singapore.

PHOTO 9. A Canadian Robo-One with a PDA brain.

PHOTO 10. Contemplating robotic track and field.

PHOTO 11. Team Austria and Team Germany.

PHOTO 12. Cute li'l NanoSot robots!

PHOTO 13. Team Korea gets ready for a MiroSot match.

PHOTO 14. A RoboSot bot shoots and scores!

PHOTO 15. Bot's eye view.

In the summer of 2003, I decided to start a robotics club in Boise (apparently several other people had the same idea). So, I consulted with the Discovery Center of Idaho (a local hands-on science museum) and placed notices in two local newspapers announcing the creation of the Boise Robotics Group — the BoRG.

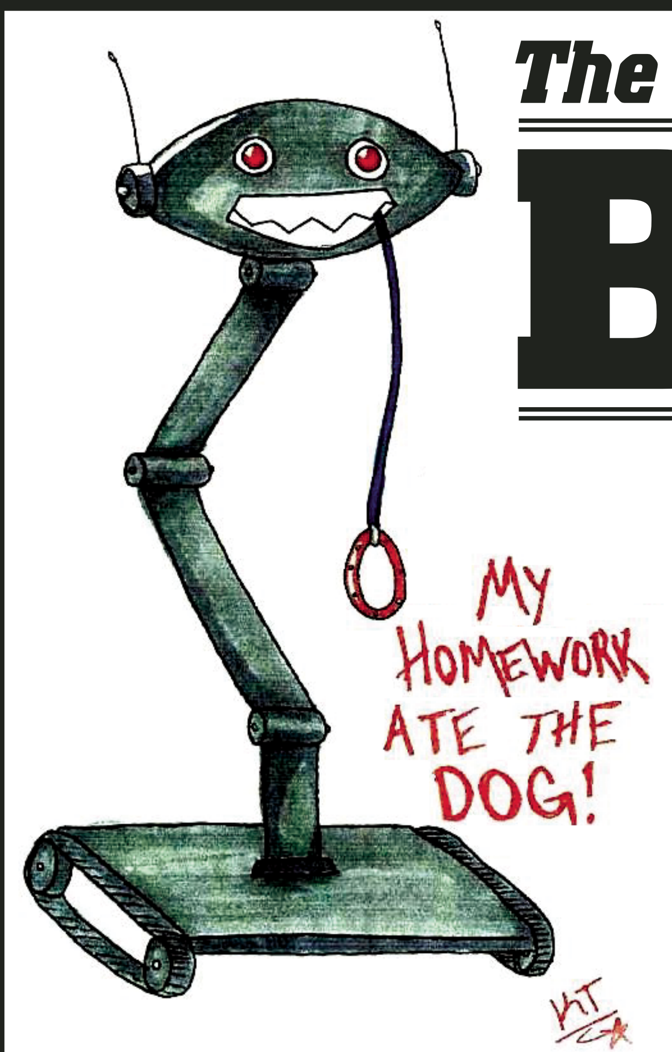
Our first meeting took place in January 2004 at the Discovery Center, and I was delighted that more people attended the kick-off meeting than I expected. Since then, we've held classes in basic electronics, Stamp programming, printed circuit board (PCB) design and construction, and soldering. We have a logo, hold competitions, and are writing a robotics book for our new members. We're still a small group, but new people show up at most meetings. That's not bad for a robotics group that's barely two years old.

In this article, I'd like to tell you about seven of the projects that the BoRG is currently working on and how we're accomplishing them (and how you can, also!). Perhaps our experiences will generate interest in some readers. This article is laid out pretty much in the order in which the BoRG developed these projects. If you're interested in any of our projects, please feel free to contact us. We can make arrangements to send some materials.

The BoRG Board Version 1

Many microcontroller enthusiasts have cut their teeth on the BASIC Stamp 2. When plugged into the Board of Education (BOE) and combined with Parallax's *What is a Microcontroller* book, it's a powerful learning platform. Since I wanted members of the BoRG Collective to learn PCB construction and soldering, along with programming the BASIC Stamp, I designed the BoRG Board — the Collective's learning platform. So far, every BoRG member has begun their robotics experience by first shooting and building their own PCB.

The BoRG Board has a 2" x 6" breadboard and 16 I/O ports configured like servo connectors (I/O, +5V, and ground). Having a large breadboard makes circuit design easier. Afterwards, if a BoRG member is happy with the performance of their



The BoRG

The Boise Robotics Group

by L. Paul Verhage

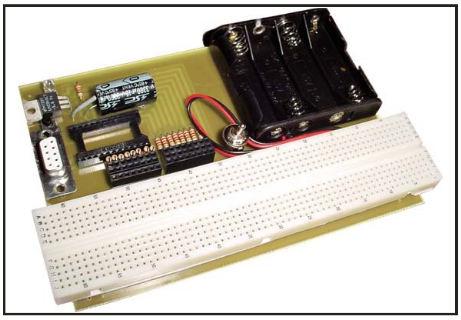


FIGURE 1. A completed BoRG Board. The PCB is shot and etched by its owner. After drilling the board, BoRG members solder components to their board.

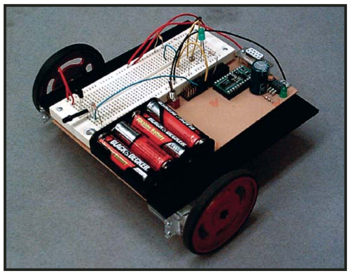


FIGURE 2. The BoRG Board is not a dead-end project. The BoRG Board becomes a robot after mounting it to a six inch square of Sintra and a pair of servos and wheels.

FIGURE 3. The alkaline battery screwdriver as purchased from our local Walmart and after being assimilated by the BoRG.

finished circuit, they can transfer it to a perf board (or PCB) and terminate it with the BoRG standard three pin header. Now their circuit plugs anywhere on the BoRG Board's expansion port.

Cheap Motors

After about nine months, the BoRG spent time searching for cheap and locally available motors to assimilate. The motors needed to have a reasonable amount of torque, so we searched for gear head motors. There are many solutions out there, but some of them are quite expensive. Here's the solution the BoRG discovered — a $7.87 alkaline screwdriver from Walmart.

The screwdrivers contain a hefty motor that draws 0.5 amps running and 1.9 amps stalled. The drive gears are configured in a planetary gear train inside the nose of the screwdriver.

Here's how the BoRG assimilates electric screwdrivers.

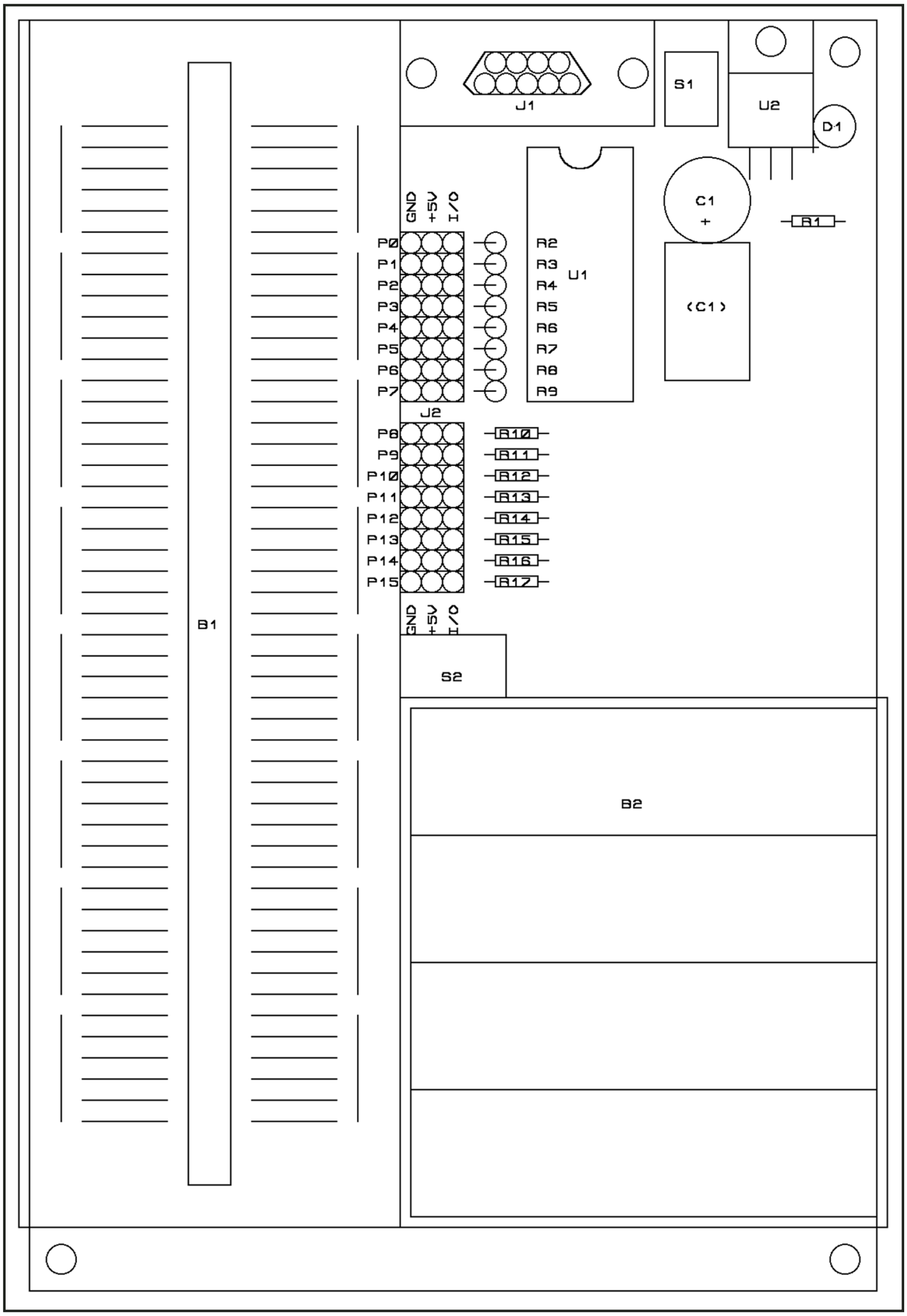
- Pull the battery holder out of the screwdriver handle. (We've tried reusing the battery holder, but its odd shape makes it difficult to mount.)

- Use an Exacto saw or hack saw and cut the case in half, just before the metal contacts in the motor (don't cut the motor).

- Remove the U-shaped pin holding the gear train housing to the motor housing.

- Carefully pull the gear train hous-

FIGURE 4. The placement of parts on the BoRG Board.



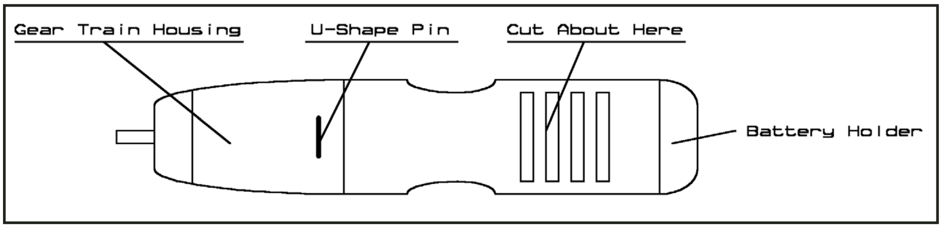


FIGURE 5. Where to cut the screwdriver and to find the U-shaped pin.

ing from the screwdriver case. (The gears can fall out of the housing, however, it's easy, but slightly greasy, to replace them.)

• Open the screwdriver case along its seam.

• Remove the motor and cut away the plastic and metal switch from the back of the motor.

• Solder wires to the two metal contacts on the motor.

• Put the motor back into its case and close the case.

• Snap the gear train back into place on the screwdriver. (You may have to rotate the motor drive gear to get everything to line back up.)

• Replace the U-shaped pin.

Here's how the BoRG adds an axle to the motors.

• Cover half of a one inch long headless 1/4-20 setscrew in heat shrink. (We purchased the setscrew from Ace Hardware — it's part number M71567 and is located in their small parts boxes.)

• Pound the covered half of the setscrew into the screwdriver chuck.

Alternatively, you can weld the setscrew to the screwdriver, but be sure the set screw remains true to the chuck during the weld.

One and one quarter inch (1-1/4") galvanized steel pipe straps are perfect for mounting the screwdrivers to a robot base. The pipe straps are available in a bag of four from Lowe's home improvement store for $1.43, or $0.35 a piece. The photo of the cutting board robot illustrates how to mount the screwdriver motors to a robot base.

Our Favorite H-bridge

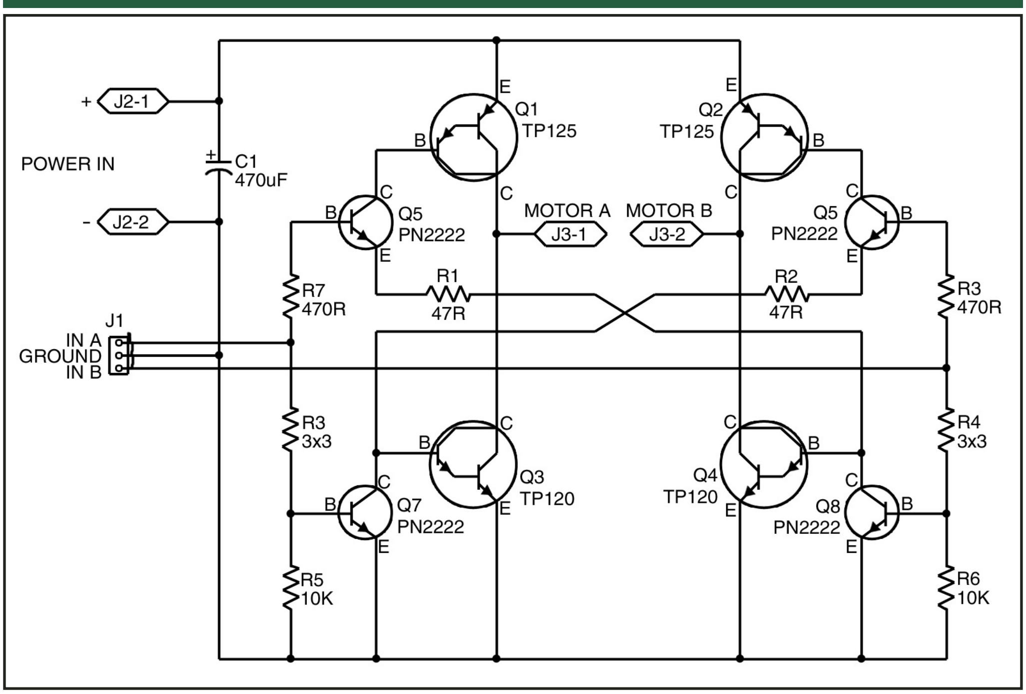
For the BoRG who are ready to use screwdriver motors, we developed an H-bridge PCB based on Bob Blick's H-bridge design. I discovered Bob Blick's H-bridge in 2000, while doing research for my Basic Brick — a BS-2 version of MIT's Programmable Brick. I recommended the BoRG assimilate the Bob Blick design, since the H-bridge uses readily available components and is suitable for motors in small- and medium-sized robots. You can find information on Bob's design at his website: **www.bobblick.com/**

The BoRG H-bridge PCB is a dual H-bridge design and terminates in a 2 x 6 header that plugs into the BoRG Board expansion port. A second header, 1 x 3, plugs into the motor's battery.

Big Wheels on the Cheap

Once we had an H-bridge and motor we were happy with, the BoRG needed to assimilate some wheels. We discovered some great wheels available in two, three, or four inch diameters. Best of all, they cost less than a dollar a piece. Of course, they needed some drilling, but then, how can you call yourself a robotics engineer if you don't perform mechanical engineering along with electrical and logical engineering?

FIGURE 6. The Bob Blick H-bridge. Schematic provided by Bob Blick.



Many stores carrying plumbing supplies have plastic caps (usually made from PVC, but sometimes polystyrene) with flat tops. Finding the center of the caps (where they are mounted to their axle) is easy with a jig design found several years ago in a wood working catalog.

Here's how to make the jig.

• Cut a four inch square out of 1/16 inch thick modeling plywood (cut the plywood as square as possible).

• Draw a line across one diagonal of the plywood (this will be the drawing slot).

• Cut the drawing slot out of

the center of the jig (make the slot about two inches long and 1/16 inches wide).

• Cut two pieces of 1/4 inch x 1/4 inch basswood strip 3-1/2 and four inches long (these are the alignment strips).

• Glue the alignment strips to two adjacent sides of the plywood at a 90 degree angle.

To locate the center of a cap, place the cap firmly against the sides of the jig and draw a line through the drawing slot and across the face of the wheel. Rotate the cap about 90 degrees and draw a second line across the face of the cap. The intersection of the two lines marks the center of the cap.

To mount the wheel to a screwdriver, we drill a 1/4 inch hole through the center of the cap and bolt it to the setscrew. To increase the wheel's traction, the BoRG wraps a #84 rubber band around the wheel. This rubber band is a popular size and available at office supply stores like Staples.

Cheap Robot Bases

The BoRG builds robots on several bodies. Our first robot body was a six inch square of Sintra. The BoRG Board and servos were attached to the Sintra with double-stick foamy tape. Sensors are then taped or bolted to the Sintra. Now we have two additional robot bodies in development.

The Cutting Board Robot

We wanted something reasonably large, cheap, and available for a new robot body. We found one answer in a plastic cutting board. Plastic cutting boards are made from high density polyethylene. The ones used by the BoRG measure 9 inches by 12 inches and come with a convenient handle. There's lots of real estate on a cutting board for the robot controller and a suite of sensors.

Battery-operated screwdriver motors are easily mounted to the bottom of the cutting board with pipe

FIGURE 7. Parts placement for the BoRG H-bridge.

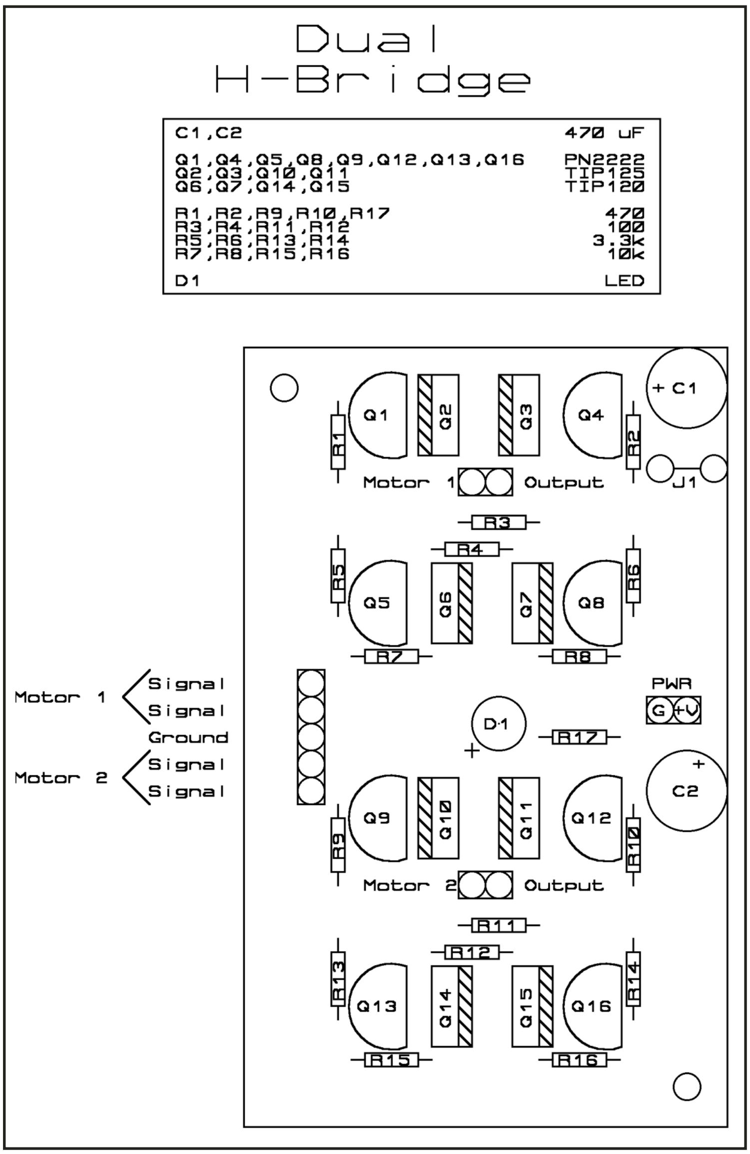


FIGURE 8. Large wheels don't come any cheaper than plumbing supplies.

FIGURE 9. This jig was made from modeling plywood and bass wood strips.

FIGURE 10. Jig diagram.

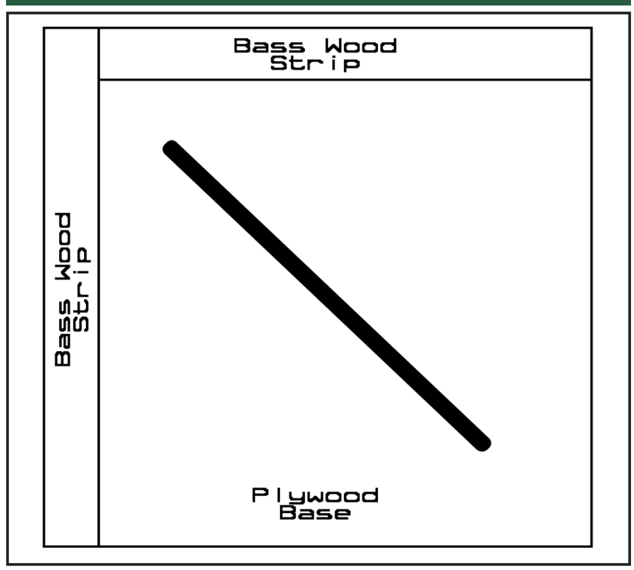
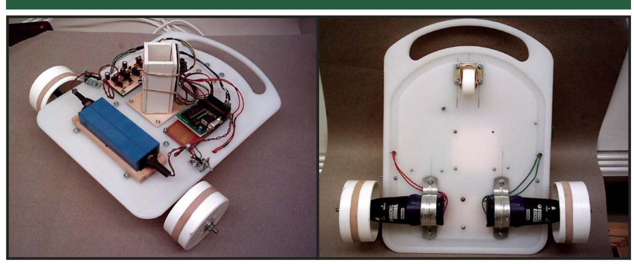


FIGURE 11. Two views of a cutting board robot base.



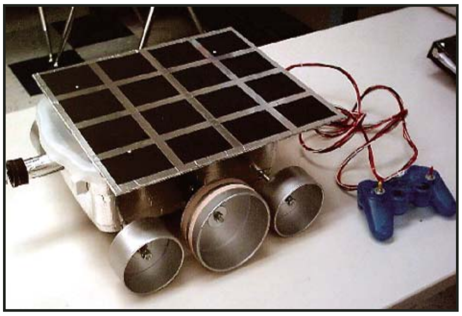


FIGURE 12. A peek at the Shoebox robot. There will be more about these robots in a future article.

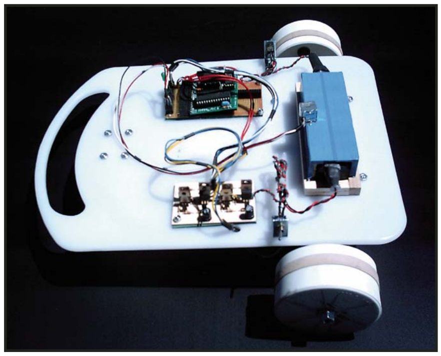


FIGURE 13. A Blade Runner test bed.

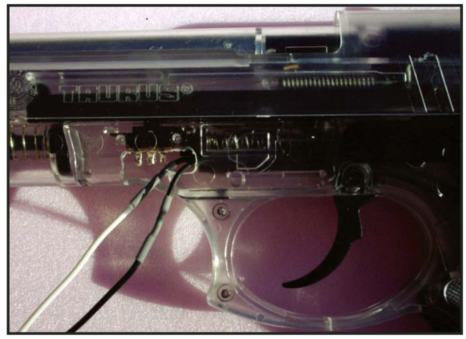


FIGURE 14. Note where the pistol was cut away to expose the switch.

straps and #6-32 nuts and bolts. Three-inch diameter plumbing caps make good drive wheels for the cutting board robot. A small swivel wheel (available at home improvement stores) is mounted in back as a tail dragger wheel.

The Shoebox Robot

Another robot base used by some of the BoRG Collective is the plastic shoebox. This BoRG project originated after I judged the rover races for the Oregon Star Party.

Unfortunately, their rovers only had a single drive motor and, therefore, could only go forwards or backwards. Contestants kicked their rovers to make them turn. Ouch! I vowed to do something about this rover abuse and came up with an inexpensive and flexible rover design that some of us are adapting to robotics.

Walmart sells a Rubbermaid 6.5 quart plastic shoe box. At $1.32, it's cheaper than a plastic cutting board and has plenty of room for motors, controllers, and sensors. It's easy to cut panels out of the shoebox by drilling the shapes out with a Dremel and drill bit. Be careful while you drill that you don't crease the shoebox and crack it. However, if you do manage to crack the shoebox, just wrap it with colorful tape when you're finished machining.

This example of a shoebox robot has three pairs of wheels and only the middle wheels are driven. Four inch wheels are used for the drive wheels and three inch wheels for the idlers. The idler wheels have their centers about 1/4 inch higher than the drive wheels. This allows the rover robot to rock a bit when turning (I found this design when investigating RC cars to modify into robots).

I'm in the process of constructing a solar power rover with a screwdriver motors, PVC caps, and a version 2 BoRG Board. (Do you find that your robot projects are growing faster than you can complete them?)

The Blade Runner Project

There's some disappointment in the BoRG Collective about BattleBots, since they're not autonomous. While there's still a lot of engineering that goes into a BattleBot (not to mention money), the BoRG Collective is more interested in watching slug-fests between autonomous robots.

To help bring about autonomous robot combat, some BoRG members are developing the Blade Runner Project. Using beacons, sensors, and Soft Air pistols, robots will be able to hunt down other robots (Replicants) and retire them. Along with modifying pistols, we're also experimenting with sensor suites and rules for the Blade Runner competition.

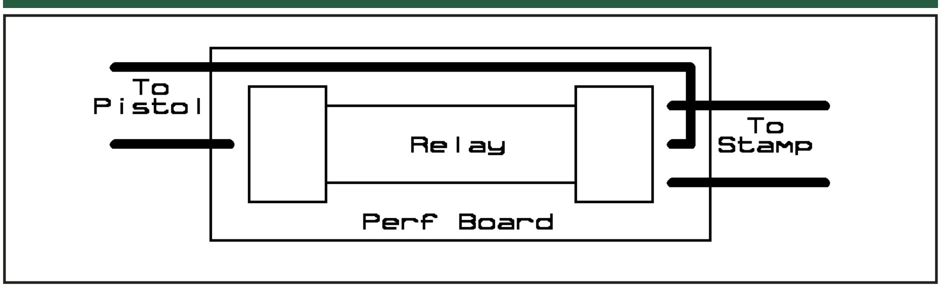
The Soft Air pistol uses a single switch (operated by pressing the trigger) to operate its motor. The motor drives a piston inside a cylinder, creating the air pressure needed to shoot its plastic pellets. By purchasing the transparent Taurus PT-92 pistol, we discovered the switch's location inside the pistol. The switch is a simple SPDT monetary slide switch and a robot can fire the pistol if the switch is removed and replaced with a relay.

Normally, the pistol safety prevents the trigger from physically activating the switch. The modification described here does away with that safety, so please exercise caution when adapting the pistol to microcontroller control (the Soft Air pistol is a pretty safe pistol, but still wear safety glasses when working around one).

In Figure 14, you can see where you drill out a small portion of the pistol's body to access the power switch.

Here's the modification procedure used by the BoRG.

FIGURE 15. The perf board.



- Remove the batteries from the pistol.
- Use a Dremel and small drill bit to cut out a small square out of the pistol's plastic case (*Note: Don't drill very deep, or else you'll damage the switch and its wiring*).
- Unsolder the two wires from the switch and pull them out of the pistol.
- Extend the length of the wires by sol-

dering six additional inches of wire (cover in heat shrink afterwards).

• Zip tie the wires to the finger guard of the pistol for a strain relief.

Now assemble the relay control perf board:

• Cut a piece of perf board large enough for the five-volt reed relay (RadioShack 275-0232) and connecting wires (in this example, the board measures 0.4 inches wide and 2.2 inches long). For strain relief, enlarge some of the holes in the perf board. Pass wires through the strain relief holes before soldering them to the perf board.

• Solder the relay to the perf board.

• Solder wires from the Soft Air trigger to the relay on the perf board.

• Cut two additional wires to connect the relay to your microcontroller.

• Solder the two new wires to the perf board.

• Pass the wires through their strain relief in the perf board.

• Slide a 2.5 inch long piece of larger diameter heat shrink over the relay and its perf board.

• Shrink the tubing down and fill its open ends with hot glue.

Now, terminate the wires to your microcontroller as is appropriate for your control board. The BoRG terminates wires as follows:

• Slide thin heat shrink over the two wires.

• Strip and tin the ends of the wires.

• Tin the short ends of a male header that has been snapped three pins long.

• Solder the ends of the wires to the first and third header pin (I/O and ground).

• Cover the soldered connection to the header with heat shrink.

A Blade Runner robot uses the following code to fire the Soft Air pistol once:

```
Relay CON 7

HIGH Relay
PAUSE 150
LOW Relay
```

In this example, the relay is connected to I/O pin 7. Extending the length of the pause fires more pellets from the pistol. Because the Soft Air pistol is gravity fed, it should be kept upright. However, I successfully fired one while it was laying on its side. I wonder how irresponsible it would be for me to interface a PIR sensor to a robot carrying a Soft Air pistol?

Some Initial Ideas About the Blade Runner Competition

Both Replicant and Blade Runner will carry infrared beacons. The Replicant is programmed to flee from the beacon and the Blade Runner is programmed to chase them. Sonar could be used to gauge distances before firing the Blade Runner's pistol. We'll probably try using a cheap RC so the Replicant can signal the Blade Runner that it was struck.

We haven't designed the hit detector on the Replicant, but I'm getting ready to test a sound sensor and metal pan (thanks to an article by J. Ronald Eyton in the January 2006 *Nuts & Volts*). Since the impacts of Soft Air pellets are so fleeting, I plan to incorporate either a flip-flop or opto-SCR to record the hit.

The next issue to tackle is the design of the playing field. Perhaps a cityscape can be designed using plastic boxes to represent the buildings. Their placement would create the obstacle course that the Replicant and Blade Runner chase through. Roads could be elevated in places with the use of ramps. The elevated roads add an additional challenge, as now the robots must ensure they don't drive off the road.

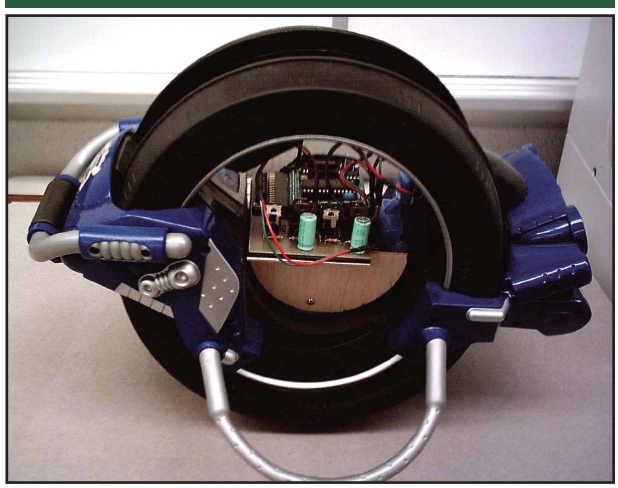
The BoRG Board Version 2

We have a more compact, second generation robot controller: the BoRG Board Version 2 (V2.0). The BoRG Board V2.0 unites the Parallax BS2-OEM and Bob Blick's H-bridge. Using the BS2-OEM makes the board less expensive than the version 1 BoRG Board. But the BoRG Board V2.0 doesn't contain a large breadboard, so it's not suitable for training new BoRG members how to breadboard and interface electronics. The BoRG Board V2.0 uses the same style I/O port found in the first BoRG Board. This robot board is in the early stages of testing, but so far, has worked well in the robots it was tested in.

The BoRG Book

Since inexperienced people join the BoRG all throughout the year, we've decided to write an introductory robotics book. Not that there aren't good robotics books on the market already — there's a ton of them. We're writing the BoRG Book as an inexpensive introduction to the way the BoRG does robotics and the tools we use. So

FIGURE 16. A BoRG Board V2.0 being tested in a new BoRG robot.



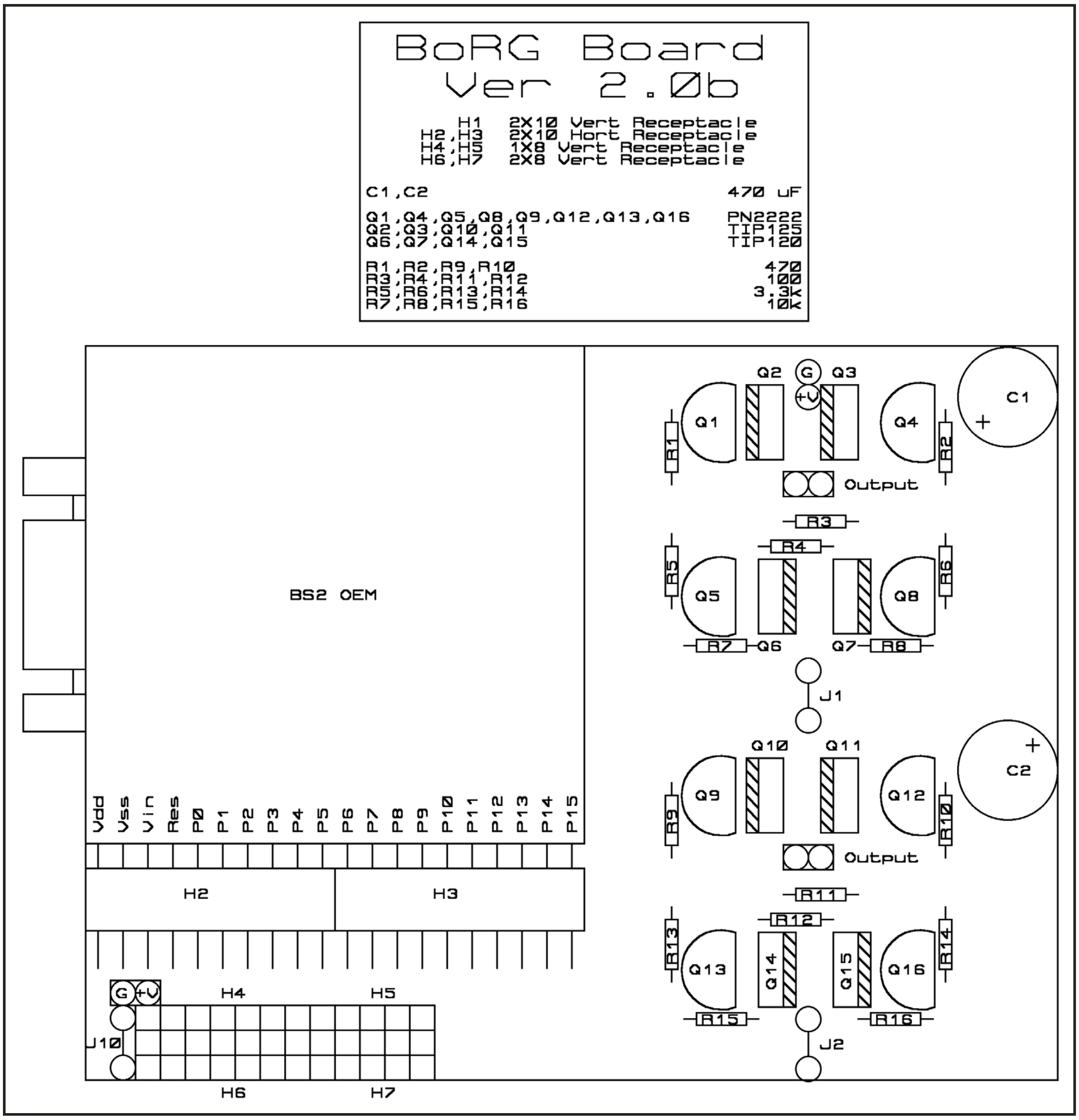


FIGURE 17. The placement of components on the BoRG Board V2.0.

far, an estimated one half of the book has been written.

When completed, each new member will get a copy of the book. With the book and help from an experienced BoRG member, newly assimilated BoRGs will learn the book's content through a series of readings, exercises, and quizzes.

After completing the book, new members will have learned basic electronics, built a PCB, assembled their first robot, and be ready to compete in competitions. There should be enough information in the book that members will be ready to tackle almost any project independently.

We feel the BoRG can support members with a wide variety of knowledge levels this way. The chapters planned for the book include the following:

An Overview
1. Schematics And Component Identification
2. Basic Electronics Skills
3. Assembling The BoRG Board
4. Testing The BoRG Board
5. Learning To Program And Breadboard On The BoRG Board
6. Assembling The BoRGBot
7. BoRGBot Competitions

A Servo Controller for the BoRG

James Cahoon — secretary of the BoRG — is experimenting with PIC microcontrollers. His first project is designing a servo controller to take some of the work load off of the BoRG Board. His inexpensive design controls up to three servos based on commands sent to it over an I2C bus. Two servo controllers can be linked together to operate up to six servos.

The first design was breadboarded, but now the BoRG has a PCB design. This servo controller could be used in place of an H-bridge for servo

FIGURE 18. Servo controller schematic. Schematic by James Cahoon.

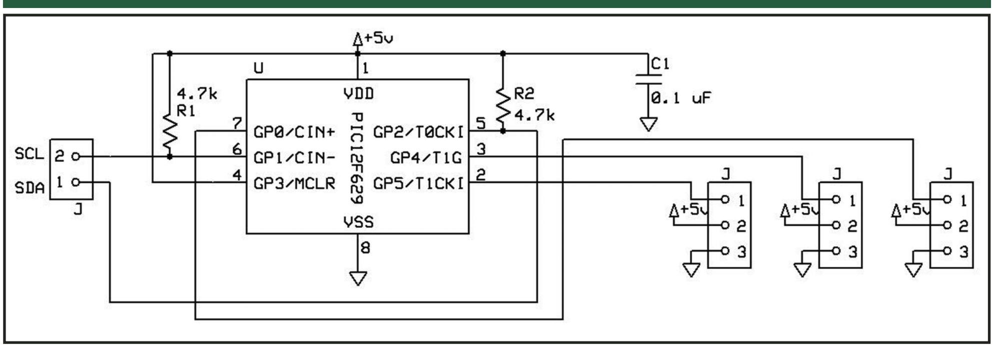
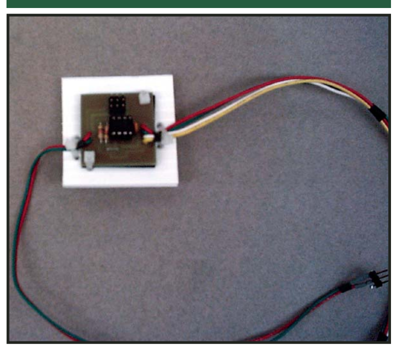


FIGURE 19. Servo controller.



modified motors, or to operate sensors mounted to the top of servo controlled masts.

Our last project is just in the planning stages, but should have already been announced by the time this article goes to print. On October 14, 2006, the BoRG plans to host a robotics competition called Lunar Ice 2006. The competition pits robots against one another as they attempt to retrieve as much ice from the lunar soil as they can. The competition places an importance on collecting as little contaminating material (rocks and sand) with the ice and doing it in the shortest possible time. We hope to make this a regular competition that will put Boise on the robotics map.

Well, that about wraps it up for the BoRG for now. Please feel free to use the copper masks in this article to shoot your own boards. And please don't hesitate to contact us with any questions that you may have. You'll find us on the Internet at **www.boiseroboticsgroup.org** If you would like to send us an email, we currently share email on Yahoo Groups under the group, BO_RG. You can also email the current president at paul.verhage@boiseschools.org. Our first competition was a success, so I've included a photograph of the contestants and their humans for your viewing pleasure. **SV**

PCB patterns/copper masks for this article can be found on the *SERVO* website at www.servomagazine.com

FIGURE 20. Placement of parts for the servo controller.

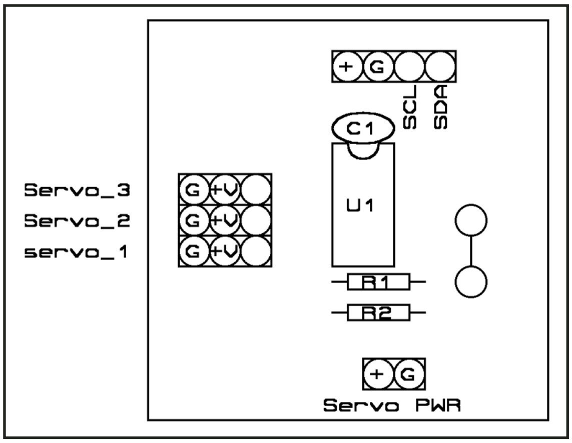
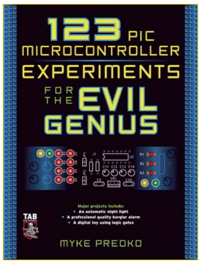


FIGURE 21. Participants of the Fall 2004 BoRG Competition.

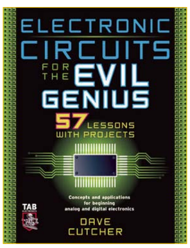
The SERVO Store

123 PIC Microcontroller Experiments for the Evil Genius
by Myke Predko



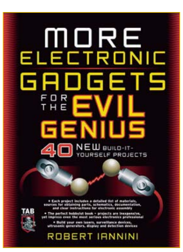
Few books take advantage of all the work done by Microchip. *123 PIC Microcontroller Experiments for the Evil Genius* uses the best parts, and does not become dependent on one tool type or version, to accommodate the widest audience possible. Building on the success of *123 Robotics Experiments for the Evil Genius*, as well as the unbelievable sales history of *Programming and Customizing the PIC Microcontroller*, this book will combine the format of the evil genius title with the following of the microcontroller audience for a sure-fire hit. **$24.95**

Electronic Circuits for the Evil Genius
by Dave Cutcher



Cutcher's 57 lessons build on each other and add up to projects that are fun and practical. The reader gains valuable experience in circuit construction and design and in learning to test, modify, and observe results. Bonus website **www.books.mcgraw-hill com/authors/cutcher** provides animations, answers to worksheet problems, links to other resources, WAV files to be used as frequency generators, and freeware to apply your PC as an oscilloscope. **$24.95**

MORE Electronic Gadgets for the Evil Genius
by Robert E. Iannini

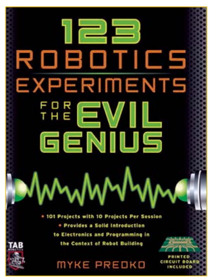


This much anticipated follow-up to the wildly popular cult classic *Electronic Gadgets for the Evil Genius* gives basement experimenters 40 all-new projects to tinker with. Following the tried-and-true Evil Genius Series format, each project includes a detailed list of materials, sources for parts, schematics, documentation, and lots of clear, well-illustrated instructions for easy assembly. Readers will also get a quick briefing on mathematical theory and a simple explanation of operation. **$24.95**

Mechatronics for the Evil Genius
by Newton C. Braga

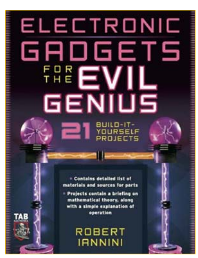
If you're fascinated by electronics and mechanics, this hands-on tour of the junction where they meet will bring you hours of fun and learning. Noted electronics author Newton Braga's *Mechatronics for the Evil Genius* guides you step-by-step through 25 complete, intriguing, yet inexpensive projects developed especially for this book. You will build your own mechanical race car, combat robot, ionic motor, mechatronic head, light beam remote control, and 20 other entertaining learning projects that take you to the heart of mechatronics. **$24.95**

123 Robotics Experiments for the Evil Genius
by Myke Predko



If you enjoy tinkering in your workshop and have a fascination for robotics, you'll have hours of fun working through the 123 experiments found in this innovative project book. More than just an enjoyable way to spend time, these exciting experiments also provide a solid grounding in robotics, electronics, and programming. Each experiment builds on the skills acquired in those before it so you develop a hands-on, nuts-and-bolts understanding of robotics — from the ground up. **$25.00**

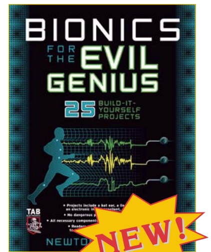
Electronic Gadgets for the Evil Genius
by Robert Iannini



The do-it-yourself hobbyist market — particularly in the area of electronics — is hotter than ever. This book gives the "evil genius" loads of projects to delve into, from an ultrasonic microphone to a body heat detector, all the way to a *Star Wars* Light Saber. This book makes creating these devices fun, inexpensive, and easy. **$24.95**

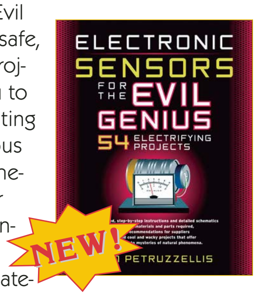
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Bionics for the Evil Genius
by Newton C. Braga



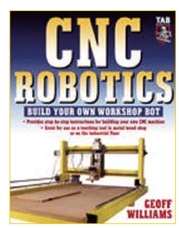
Step into the future (or the past, if you have a touch of Dr. Frankenstein in your soul) with these 25 incredibly cool bionic experiments! Demonstrating how life forms can be enhanced, combined, manipulated, and measured with electronic and mechanical components, these inexpensive projects from internationally renowned electronics guru Newton Braga provide hours of fun and learning. Totally safe, *Bionics for the Evil Genius* guides you step-by-step through 25 complete, intriguing, and low-cost projects developed especially for this book — including an electric fish, a bat ear, a lie detector, an electronic nerve stimulator, a panic generator, and 20 other exciting bioelectric/mechanical projects! **$24.95**

Electronic Sensors for the Evil Genius — 54 Electrifying Projects
by Thomas Petruzzellis



Nature meets the Evil Genius via 54 fun, safe, and inexpensive projects that allow you to explore the fascinating and often mysterious world of natural phenomena using your own home-built sensors. Each project includes a list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions. Projects include rain detector, air pressure sensor, cloud chamber, lightning detector, electronic gas sniffer, seismograph, radiation detector, and much more! **$24.95**

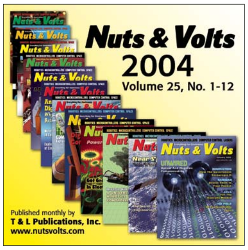
CNC Robotics
by Geoff Williams



Now, for the first time, you can get complete directions for building a CNC workshop bot for a total cost of around $1,500.00. *CNC Robotics* gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot and can be customized to suit your purposes exactly, because you designed it. **$34.95**

Nuts & Volts CD-Rom

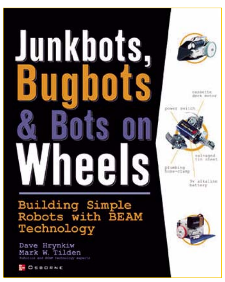
Here's some good news for *Nuts & Volts* readers! Starting with the January 2004 issue of *Nuts & Volts*, all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 25, issues 1-12, for a total of 12 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**



JunkBots, Bugbots, and Bots on Wheels

by Dave Hrynkiw / Mark W. Tilden

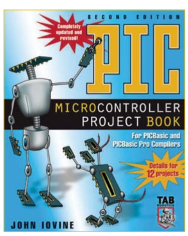
From the publishers of *BattleBots: The Official Guide* comes this do-it-yourself guide to BEAM (Biology, Electronics, Aesthetics, Mechanics) robots. They're cheap, simple, and can be built by beginners in just a few hours, with help from this expert guide complete with full-color photos. Get ready for some dumpster-diving! Get step-by-step instructions from the Junkbot masters for creating simple and fun self-guiding robots safely and easily. **$24.95**



PIC Microcontroller Project Book

by John Iovine

The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, many users of the PIC are now comfortable paying the $250.00 price for the Professional version of the PIC Basic. This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which one serves better in different situations. **$29.95**



Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

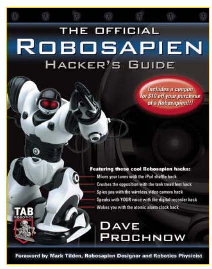
SERVO CD-Rom

Are you ready for some good news? Starting with the first *SERVO Magazine* issue — November 2003 — all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 1, issues 11-12 and Volume 2, issues 1-12, for a total of 14 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**

The Official Robosapien Hacker's Guide

by Dave Prochnow

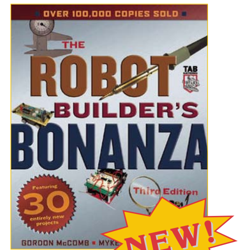
The Robosapien robot was one of the most popular hobbyist gifts of the 2004 holiday season, selling approximately 1.5 million units at major retail outlets. The brief manual accompanying the robot covered only basic movements and maneuvers — the robot's real power and potential remain undiscovered by most owners — until now! This is the official Robosapien guide — endorsed by WowWee (the manufacturer) and Mark Tilden (the designer). This timely book covers possible design additions, programming possibilities, and "hacks" not found any place else. **$24.95**



Robot Builder's Bonanza Third Edition

by Gordon McComb / Myke Predko

Everybody's favorite amateur robotics book is bolder and better than ever — and now features the field's "grand master" Myke Predko as the new author! Author duo McComb and Predko bring their expertise to this fully-illustrated robotics "bible" to enhance the already incomparable content on how to build — and have a universe of fun — with robots. Projects vary in complexity so everyone from novices to advanced hobbyists will find something of interest. Among the many new editions, this book features 30 completely new projects! **$27.95**

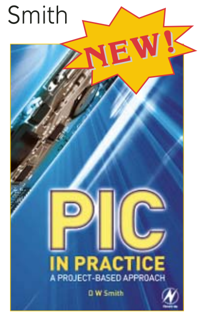


PIC in Practice

A Project-based Approach Second Edition

by David W. Smith

PIC in Practice is a graded course based around the practical use of the PIC microcontroller through project work. Principles are introduced gradually, through hands-on experience, enabling students to develop their understanding at their own pace. The book can be used at a variety of levels and the carefully graded projects make it ideal for colleges, schools, and universities. Newcomers to the PIC will find it a painless introduction, while electronics hobbyists will enjoy the practical nature of this first course in microcontrollers. **$29.95**



$20 each 15% off if you buy two or more.

From HomoSapien to RoboSapien

Before R2D2 there was R1D1

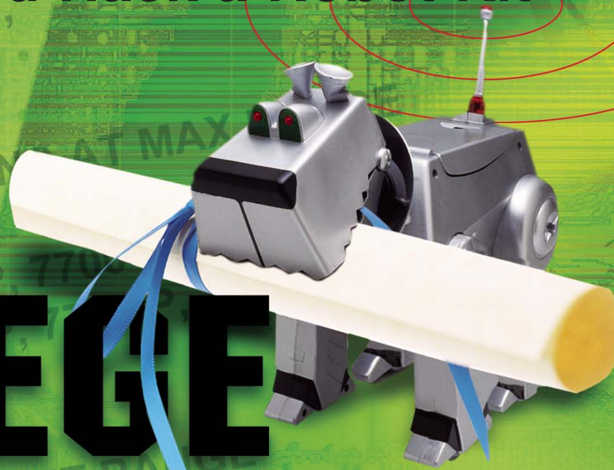
SERVO Magazine T-Shirts

Get your very own limited-edition *SERVO Magazine* T-shirt. Shirts come in sizes S, M, L, and XL, and are available in either black or white. All shirts are 100% pre shrunk cotton.

A Course on How to Build, Program, and Hack a Robot Kit

KIT KOLLEGE

WITH PROFESSOR PROCHNOW



THIS MONTH:

LECTURE 8:

JoinMax Digital Robot Dog Kit

HACKS AND MODS

JoinMax Robot Dog

An alternate hack for the JoinMax Robot Dog was demonstrated in the January 2005 edition of the "Twin Tweaks" column (see *SERVO Magazine* 01.2005).

- Substitute alkaline batteries for rechargeable batteries.
- Replace JMRD with Robopet main circuit board.
- Add Robopet IR sensor, speaker, and microphone.

During the assembly of Robot Dog Kit, changes/mods/hacks were used for streamlining the assembly process, as well as enhancing the robot's functionality.

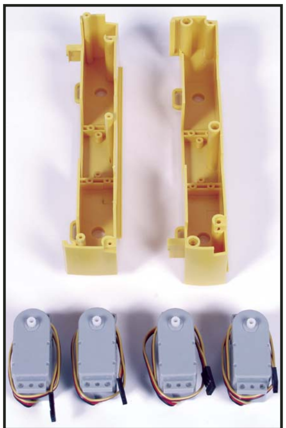
Last month's lecture introduced a simple, four-legged walking robot — the JoinMax Digital Quadruped Robot Kit. Continuing this quadruped theme, this month's Kit Kollege discusses the JoinMax Digital Robot Dog Kit (JM-DOG-001; **www.mciirobot.com** and **www.robotplayer.com**).

The Robot Dog kit is a sophisticated robot that uses 12 servo motors for providing three degrees of freedom for each leg. Two additional motors are used for animating the head and neck, while a third motor is dedicated to waging this dog's tail.

Unlike the previous JoinMax Digital kits, the Robot Dog is controlled with a unique servo motor controller circuit board (JMRD). Designed around an Atmel ATmega 16 AVR eight-bit RISC microcontroller operating at 8 MHz, the JMRD is able to drive up to 16 servo motors with a resolution of .72-degrees per step through a servo travel range up to 180 degrees.

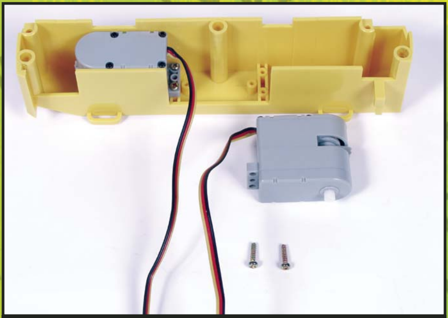
While a unique servo controller is used with the Robot Dog, the same *Mini Servo Explorer* included with all JoinMax Digital robot kits is used for programming the actions of Robot Dog. As previously mentioned in other Kit Kollege lectures regarding the Windows-only *Mini Servo Explorer*, the program creates a movement file that is played by the robot. So, unlike other robot programs, there are no logic statements in the Robot Dog program, only a series of servo power, speed, and position parameters. You are able to adjust these parameters while Robot Dog is tethered to your PC, then, after the JMRD is disconnected from your PC, these parameters are "played" by Robot Dog. Class dismissed. **SV**

STEP 1. Only a pair of diagonal cutters and a No. 0 Phillips screwdriver are needed for assembling the JoinMax Digital Robot Dog kit.

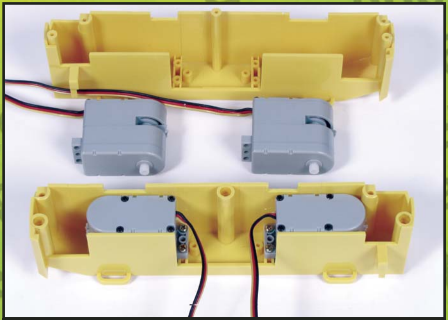


STEP 2. Four hip servo motors are installed inside the body.

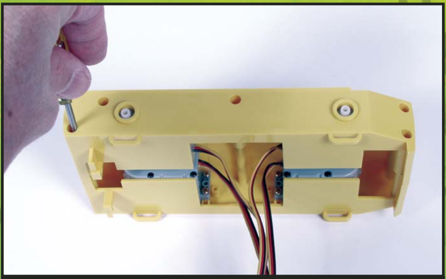
STEP 3. Use the diagonal cutters to snip the servo motor connector arm off of each hip motor.



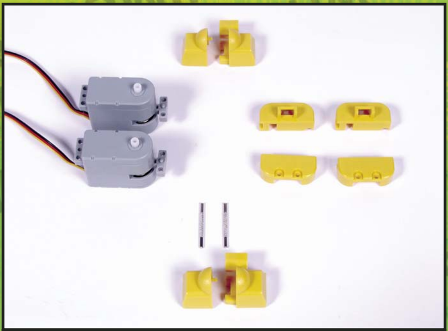
STEP 4. Two hip motors are installed in the left body half.



STEP 5. Two other hip motors are installed in the right body half.

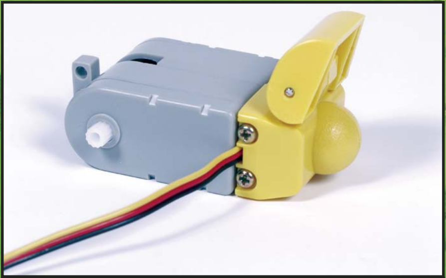


STEP 6. The left and right body halves are joined together with three screws. Two screws are inserted into the two outside lower holes and the final screw is inserted into the middle hole.

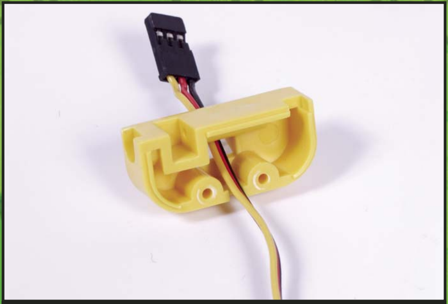


STEP 7. Each leg is built from two servos, two knee pieces, two ankle pieces, and one foot piece.

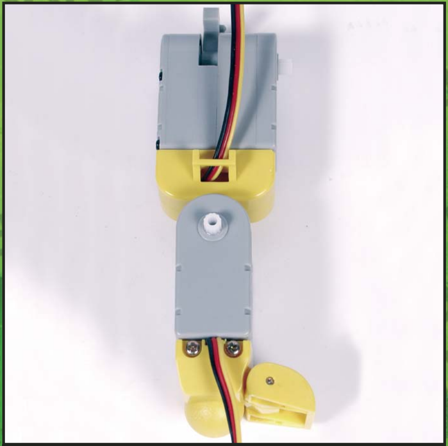
STEP 8. The left and right ankle pieces are labeled on the inside (e.g., L1, L2, R1, and R2). Make sure that you mate the two left pieces together and the two right pieces together. The foot is attached to each ankle with an axis pin. You might need needle-nosed pliers for seating this pin.



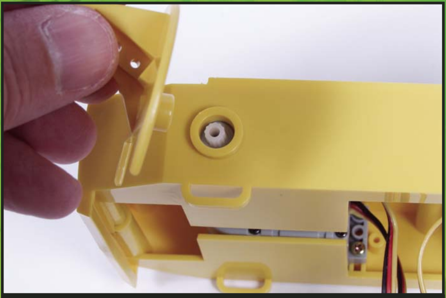
STEP 9. Make sure that each servo motor cable is routed through the inside surface of each ankle assembly.



STEP 10. Route the cable from each knee servo motor through the inside knee piece.

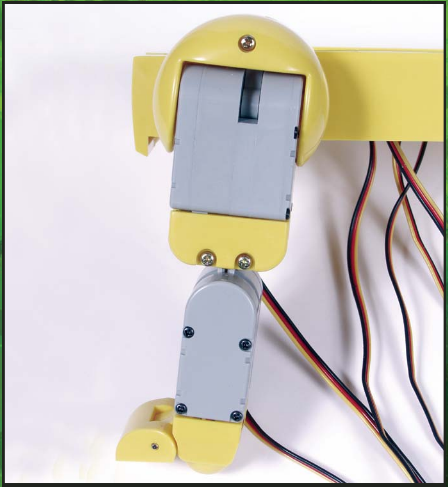


STEP 11. A completed left front leg.

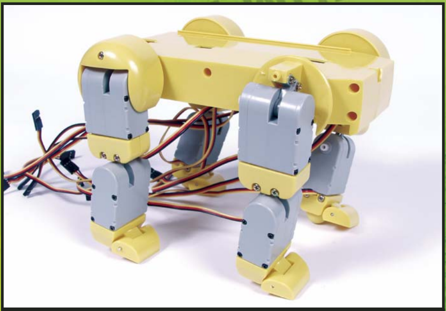


STEP 12. Attach the left front shoulder to the body.

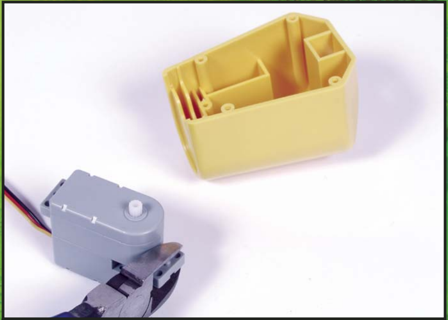
STEP 13. Mount the left front leg knee servo motor connector arm to the left front shoulder.



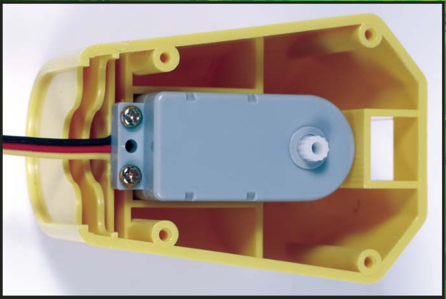
STEP 14. Trim the left front leg assembly with the left front shoulder cover.



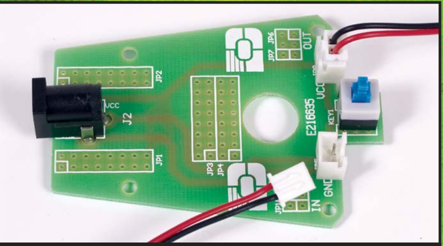
STEP 15. Assemble and attach the remaining three legs and shoulders.



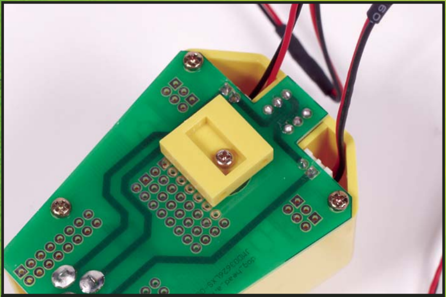
STEP 16. Begin the head assembly, by first snipping the servo motor connector arm off of the head servo.



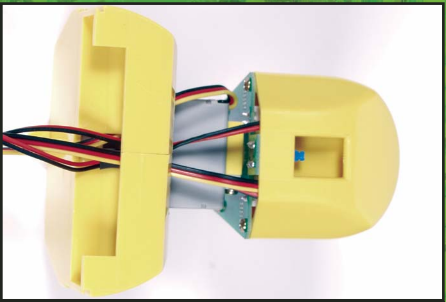
STEP 17. Route the head's servo cable out through the Robot Dog snout. This cable will later be routed back inside the head for insertion into port 1 of the JMRD.



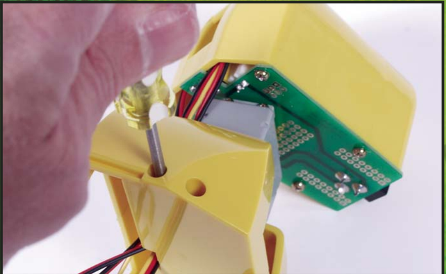
STEP 18. Insert the two battery connector cables into the two ports on the head circuit board.



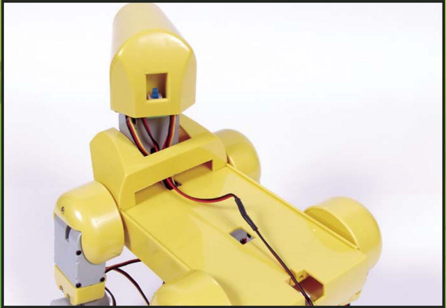
STEP 19. Route the head servo cable and the battery connector cables through the rear of the head and attach the head circuit board to the head.



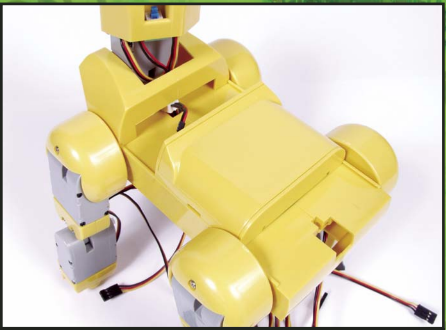
STEP 20. Route the head's cables through the neck.



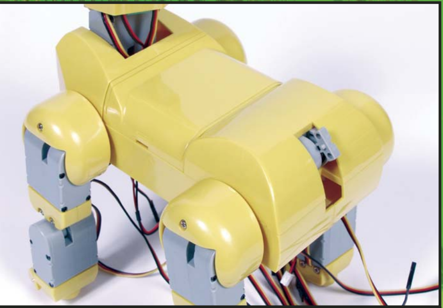
STEP 21. Attach the neck servo motor to the neck.



STEP 22. Mount the neck/head assembly on the body. This assembly slides onto the body from the front.

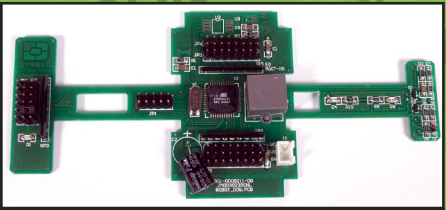


STEP 23. Slide the battery box onto the body from the rear.

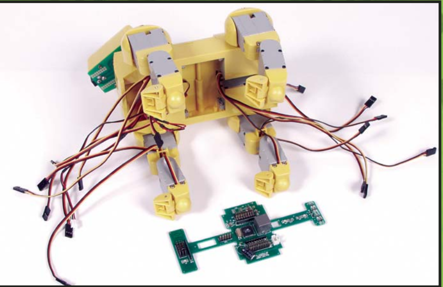


STEP 24. Slide the tail assembly onto the rear of the body.

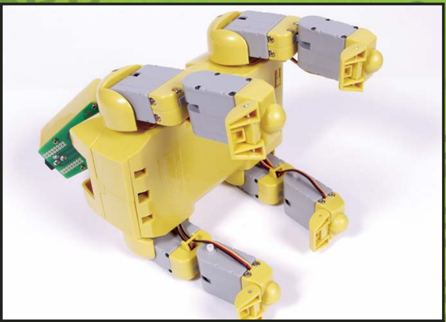
STEP 25. Fix the tail assembly and the neck/head assembly to the body with two screws (i.e., one for the tail and one for the neck/head).



STEP 26. The Robot Dog is controlled with the JMRD circuit board. This controller board has 16 I/O ports for controlling the robot's 15 servo motors (one I/O port is unused).



STEP 27. The JMRD is inserted into the underside of the Robot Dog. Each of the servo motor cables are routed through this board and plugged into an appropriate port. The head, neck, shoulder, knee, and ankle servo motor cables for the right side of the robot are plugged into ports 1 through 8. The tail, shoulder, knee, and ankle servo motor cables for the left side are plugged into ports 10 through 16.



STEP 28. Snap the bottom cover trim plate into place.

STEP 29. Attach the tail to the tail motor servo connector arm.

ROBOT KIT SOURCES

You can purchase Robot Dog from any of the following sources. Please refer to each website for updated pricing information.

British Robotics
www.britishrobotics.com

Cyber Hardware & Software
www.cyberhs.it

Garage Technologies, Inc.
www.garage-technologies.com

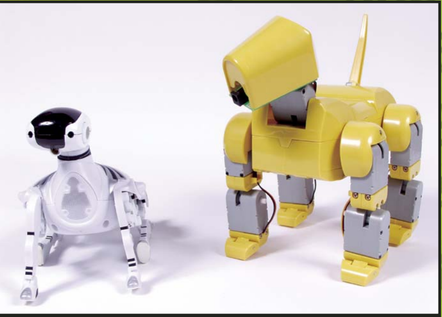
MCII Robot
www.mciirobot.com

Pololu Corporation
www.pololu.com

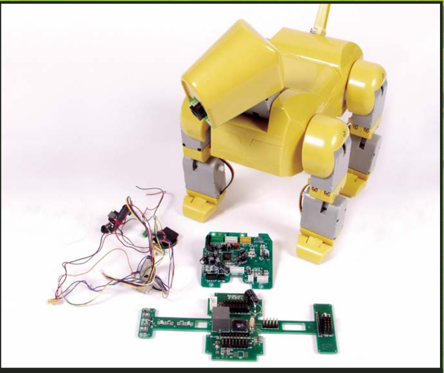
AUTHOR BIO

Dave Prochnow is a frequent contributor to Nuts & Volts and SERVO Magazine, as well as the author of 26 nonfiction books including the mega-hit The Official Robosapien Hacker's Guide (McGraw-Hill, 2006) and the upcoming PSP Hacks, Mods, and Expansions (McGraw-Hill, 2006). Dave also won the 2001 Maggie Award for the best "how-to" article in a consumer magazine. You can learn more about this Robosapien book and other robotics/electronics projects at Dave's website: ***www.pco2go.com***

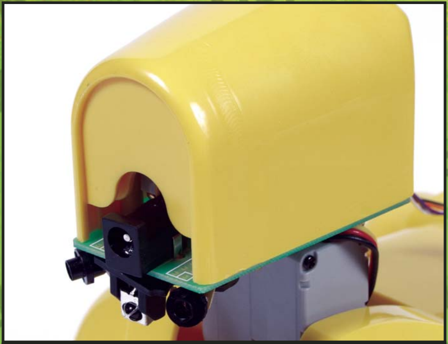
STEP 30. The completed Robot Dog kit.



STEP 31. The Robot Dog is a sophisticated quadruped that can take advantage of some of WowWee Robotics Robopet sensors.

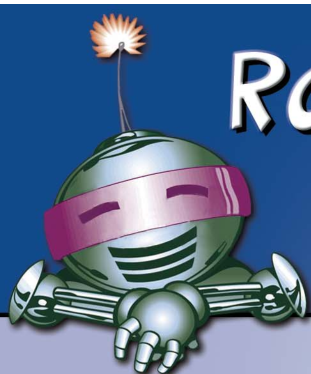


STEP 32. If you elect to hack Robot Dog with Robopet's main circuit board, you will also have to rewire each shoulder servo. Remember, Robopet does not have servo control for knee and ankle motors.



STEP 33. You can readily attach Robopet's IR sensor system to the underside of Robot Dog's head.

STEP 34. Hacking Robot Dog with Robopet's main circuit board gives you one distinct advantage over the stock JoinMax kit — the ability to use the IR remote control for programming Robot Dog.



Robotics Resources

Tune in each month for a heads-up on where to get all of your "robotics resources" for the best prices!

By Gordon McComb

Look, This Robot Can See!

Video and Imaging Technologies for Robotics

For a robot to be truly useful, it needs senses — the more senses, the better. It's easy to endow even the most basic robot with a sense of touch — all it takes is a couple of small switches. Non-contact "touch" sensors are only slightly more complicated. In addition to building our own, we now enjoy a wide assortment of low-cost commercially available ultrasonic and infrared proximity detectors.

But the one sense that offers the most potential to the science of robotics is also the elusive — vision. Robotic sight is something of a paradox: the basic sensor mechanism for providing a video image is actually quite mundane, even more affordable that many ultrasonic and infrared detectors we rely on. You can now purchase black-and-white video cameras for under $30, complete with lens. Most operate under 12 volt battery power, and can "see in the dark" because their sensor element reacts to infrared light.

There's a second part to the vision equation — what to do with the image data once it's been acquired. There is a burgeoning science of "machine vision" that seeks to provide answers. So far, most robotic vision systems are limited to controlled lighting and limited object recognition. But, as her recent series of articles in *SERVO* attest, author Robin Hewitt has shown that there's plenty of promise using even an ordinary web cam and some Windows-based software.

In this column, we'll look at some options for robotic vision, including ready-made systems that don't require a Windows, Linux, or other desktop/laptop computer to operate. We'll limit ourselves to the low-end of the spectrum — with $20,000 you can purchase a robust machine vision system, but this is a wee bit higher than most of us what to spend.

Simple Sensors for Sight

To kick off our discussion, "vision" doesn't always mean a full video-like representation of a scene. A number of simple electronic devices can be used as eyes for your robot. These include:

• *Photoresistors*, typically a cadmium sulfide (CdS) cell (often referred to simply as a *photocell*). A CdS cell acts like a light-dependent resistor — the resistance of the cell varies depending on the intensity of the light striking it. When no light strikes the cell, the device exhibits very high resistance, typically in the high hundreds of thousands of ohms, or even megohms. Light reduces the resistance, usually significantly (a few hundreds or thousands of ohms). CdS cells are very easy to interface to other electronics, but they are somewhat slow reacting, and are unable to discern when light flashes more than 20 or 30 times per second. This trait actually comes in handy, as it means CdS cells basically ignore the on/off flashes of AC-operated lights.

• *Phototransistors* are very much like regular transistors, with their metal or plastic top removed. A glass or plastic cover protects the delicate transistor substrate inside. Unlike CdS cells, phototransistors are very quick acting, able to sense tens of thousands of flashes of light per second. The output of a phototransistor is not "linear," that is, there is a disproportionate change in the output of a phototransistor as more and more light strikes it. A phototransistor can become easily "swamped" with too much light. Even as more light shines on the device, the phototransistor is not able to detect any more change.

• *Photodiodes* are the simpler diode versions of phototransistors. Like phototransistors, they are made with a glass or plastic cover to protect the semiconductor material inside them. And, like phototransistors, photodiodes are very fast acting, and can become "swamped" when exposed to a certain threshold of light. One common characteristic of most photodiodes is that their output is rather low, even when fully exposed to bright light. This means that to be effective, the output of the photodiode must usually be connected to a small amplifier of some type.

Photoresistors, photodiodes, and phototransistors are connected to other electronics in about the same way — a resistor is placed between the device and either +V or ground. The

point between the device and the resistor is the output. With this arrangement, all three devices, therefore, output a varying voltage. The exact arrangement of the connection determines if the voltage output increases or decreases when more light strikes the sensor.

Light-sensitive devices differ in their spectral response — the span of the visible and near-infrared light region of the electromagnetic spectrum that they are most sensitive to. CdS cells exhibit a spectral response very close to that of the human eye, with the greatest degree of sensitivity in the green or yellow-green region. Both phototransistors and photodiodes have peak spectral responses in the infrared and near-infrared regions. In addition, some phototransistors and photodiodes incorporate optical filtration to decrease their sensitivity to the visible light spectrum. This filtration makes the sensors more sensitive to infrared and near-infrared light.

Eyes From Static CMOS Memory

Long before solid-state (CCD and CMOS) camcorders and digital cameras became common, robot experimenters used to play around with static CMOS RAM (Random Access Memory), using modified chips as multi-cell eyes for their creations. Most all semiconductors are sensitive to light, even the storage matrix inside memory chips. By using static memory, the interface to the chip can be kept simple and straight-forward. In fact, all you need to do is connect some wires from the chip to your computer or microcontroller.

You can sometimes (stress on the "sometimes") find static CMOS memory chips already modified for use as vision sensors. But if not, you can make one yourself. Chips like the 2114L3 memory IC are no longer made, but you can sometimes find them surplus, especially from outfits that specialize in parts for refurbishing video arcade games and older single-board computers. Adding to the challenge is that you need to find the kind in a ceramic case, outfitted with a soldered metal lid.

To begin, you'll need to get the metal lid off in order to expose the semiconductor die inside the chip. The best way to do this is with a small butane torch. This can be tricky — and it's dangerous — so proceed with caution. Secure the chip in a metal vise, and *carefully* apply even flame over the lid. After 5-10 seconds, the solder should melt. Quickly remove the flame, and slip the lid *all the way* off. Use care not to disturb the die or the connections inside the chip, or you'll ruin it.

Before you get excited about how cheaply you might be able to create your own vision system for your robot, consider that most static RAM chips suffer from rather poor sensitivity to light (it's not what they were designed for), so don't expect low-light use! In addition, the layout of the memory on the die isn't always conducive to image display. The 2114, for example, is laid out in four separate quadrants, rather than one large matrix.

If you're interested in experimenting with static RAM vision, check out **Futurbots.com** In addition to surplus inventory that includes the 2114, they offer the Mostek 4008, along with a reprint of a mid-1970's *Popular Electronics* article about using the chip for vision.

Enhancing the View With Lenses and Filters

Lenses and filters can be used to greatly enhance the sensitivity, directionality, and effectiveness of both single- and multi-cell vision systems. By placing a lens over a small cluster of light cells, for example, you can concentrate room light to make the cells more sensitive to movement of humans and other animate objects.

Optical filters can also be used to enhance the operation of light cells. Optical filters work by allowing only certain wavelengths of light to pass through, blocking the others. CdS photocells tend to be sensitive to a wide range of visible and infrared light; you can readily accentuate the sensitivity of a certain color (and thereby de-accentuating other colors) just by putting a colored gel or other filter over the photocell.

Lenses are refractive media constructed so that light bends in a particular way. The two most important factors in selecting a lens for a given application is lens focal length and lens diameter.

- *Lens focal length*. Simply stated, the focal length of a lens is the distance from the lens where rays are brought to a common point. (Actually, this is true of "positive" lenses only; "negative" lenses behave in an almost opposite manner.) See below for more information.

- *Lens diameter*. The diameter of the lens determines its light gathering capability. The larger the lens, the more light it collects.

- *Filters*. Filters accept light at certain wavelengths and block all others. A common filter used with robot design is made to pass infrared radiation and block visible light. Such filters are commonly used in front of phototransistors and photodiodes to block out unwanted ambient (room) light. Only infrared light — from a laser diode, for instance — is allowed to pass through and strike the sensor.

The Exciting World of Video Vision

Simple light detection is useful for sensing the absence or presence of light, but it cannot make out shapes of objects. This greatly limits the environment such a robot can be placed into. The next step up is video vision, using sensors that provide a 2D pixelated view of the world. Even as recently as five years ago, a video vision was an expensive proposition for any robot experimenter. But the advent of inex-

pensive "pinhole" cameras — so called because they are used in place of the pinhole lens in the front door of a house or apartment — now makes the hardware for machine vision affordable.

A video system for robot vision need not be overly sophisticated. The resolution of the image can be as low as about 100 by 100 pixels (10,000 pixels total), though a resolution of no less than 300 by 200 pixels (60,000 pixels total) is preferred. The higher the resolution, the better the image and, therefore, the greater the ability to discern shapes. A color camera is not mandatory, though image analysis is often made easier by detecting the color of an object, rather than its shape.

Video systems that provide a digital output are generally easier to work with than those that provide only an analog video output. Digital video systems can be connected directly to a PC, typically through a USB port. Analog video systems require a video capture card, fast analog-to-digital converter, or other similar device attached to the PC.

PC-Based Vision

If your robot uses a PC motherboard or laptop, you're in luck! There's an almost unlimited array of inexpensive digital video cameras that you can attach to a PC. The proliferation of web cams for use with personal computers has brought down the cost of these devices to under $50, and often less for bare-bones models. You can use a variety of operating systems — Windows, Linux, or Macintosh — though be forewarned that not every web cam has drivers for every operating system. Be sure to check before buying.

Connecting the camera to the PC is the easy part. As mentioned above, using its output is far more difficult. You need software to interpret the video scene that the camera is capturing. However, there are plenty of examples to start from, so you don't have to code from scratch. A number of interesting and useful articles on machine vision have appeared in both *SERVO Magazine* and *Nuts & Volts*. Start with these, and then check out the sources for commercial and experimental vision software provided in the Sources list.

If you're a user of the .NET programming platform under Windows, and are fairly familiar with C# or VB programming, be sure to investigate the DirectShow.Net Sourceforge project at **directshownet.sourceforge.net** DirectShow.Net is a managed .NET wrapper that allows you to tap into the incredibly powerful DirectShow architecture of Windows, without the need to use C++. (DirectShow programming is among the most difficult Windows' apps to write.)

The project authors provide samples of capturing video, and a couple of samples demonstrate how to retrieve the bitmap of each video frame as it goes through the system. With this bitmap, you can write your own image analysis routines, such as looking for pixels of a specific color. Or, you can use fairly simple scene averaging techniques to determine if there's movement in the frame. (Note that there are several open source .NET initiatives for Linux, BSD, and other operating systems. The DirectShow.Net library will not work with these versions, as non-Windows operating systems do not provide the necessary DirectShow architecture.)

Microprocessor-Based Vision

A PC makes it easy to integrate a web cam, but it's not the only way to provide vision to your robot. Several microprocessor-based vision systems are available that work with most any robot brain, including inexpensive microcontrollers like the BASIC Stamp and BasicX.

One of the most popular of this breed is the CMUcam2 (**www.cs.cmu.edu/~cmucam2/**), which is available commercially from a number of sources, including Acroname. This device incorporates a color imager, lens, and image analysis circuitry. The CMUcam2 can track an object by color, as well as track motion using frame differencing, isolate objects using outlines, and more. It can connect to a PC or microcontroller via a TTL connection or RS-232. Retail price is under $200. Acroname offers a less expensive version intended to be directly connected to microcontrollers (no RS-232) for $170.

Another option is the AVRcam from JRobot (**www.jrobot.net**), so named because it is based on the Atmel AVR microcontroller. You can use the AVRcam with any microcontroller or PC via a serial connection. Like the CMUcam2 the AVRcam can track objects by color. The software used in the AVRcam is open source, and you can purchase an unassembled kit with camera for $99.

If you're not afraid of hacking existing hardware, there are a number of options available for adding vision to your bot. An old favorite is using the old Gameboy Color camera (the camera takes black-and-white pictures — it was made for the old Nintendo Gameboy Color hand-held console). These are still available on eBay and surplus sources for under $10 or so. With the power of Yahoo! or Google, you can find how-to articles on interfacing the camera to microcontrollers. You can see one example at **www.seattlerobotics.org/encoder/200205/gbcam.html**

Using Composite Video for Vision

In the modern digital world, it's sometimes easy to forget that analog isn't dead — not by a long shot. Most low-cost black-and-white and color security cameras provide an analog composite video signal for direct connection to a VCR or television. There are several standards used for video signals — NTSC and PAL being the two most common around the world — and you merely select a camera that supports the standard of your VCR or television.

NTSC is used in North America, Japan, and Canada, and PAL is used in most of Europe. (Technically speaking, NTSC defines a standard for providing color. The RS-170 video standard defines the basic video signal format used by NTSC. Even for a black-and-white camera, manufacturers will still often state their product is "NTSC compatible.")

In order to process video from a composite video camera, you need a means to separate out the different parts of the signal. Monochrome video is composed of two major components: sync and video signal. Low-cost integrated circuits like the LM1881 (about $3 from Digikey, Newark, and other chip distributors; use **www.findchips.com** to locate a source) allow you to apply a composite video signal and get back separate outputs for vertical and composite sync. Using a microcontroller, you can use these signals as a means to synchronize with the video information that is also being provided by the camera. You can capture a full frame of video or analyze one line of video at a time.

In order to effectively use a sync separator IC like the LM1881, you should have a fairly good understanding of how analog video works. As this is an old technology, you can check your local library for books on the subject. Even one a few decades old will give you the framework for developing custom vision projects using analog video.

Going Further

Imaging is a huge field that spans garage hacking of old toy cameras to complex vision analysis for medical diagnoses. Spend some time using a decent Internet search engine to explore the exciting world of machine vision. You might start with the following keywords, and refine your searches from there (try these with and without quotes): image capture, machine vision, robot vision, and artificial vision. And, of course, check out the following links!

Additional Sources

123securityproducts.com
www.123securityproducts.com

Black-and-white board cameras for security; video transmitters and receivers. Boasts an extremely small (but also fairly expensive) 900 MHz wireless camera/transmitter smaller than a nine-volt battery.

Acroname
www.acroname.com

Provides several versions of the CMUcam2 single-board image analysis vision system.

AVR + GameBoy™ Camera = Fun
http://pages.zoom.co.uk/andyc/camera.htm

Detailed information, circuits, and sample programming (for the Atmel AVR microcontroller) for using the Gameboy camera for crude machine vision.

CCTV Outlet
www.cctvoutlet.com

CCTV Outlet sells a selection of cameras, lenses, RF transmitters, and receivers.

Game Boy Camera Parallel Port Interface
http://geocities.com/vjkemp/gbcam.htm

How to connect a Nintendo Gameboy camera (no longer made, but still available from some quarters) to a PC parallel port. Then, how to program the PC to know how to read the data the camera is sending it. Includes circuit diagrams, how-tos, program code.

Foveon, Inc.
www.foveon.com

Foveon, Inc. is a maker of high resolution CCD color imagers. Some technical documents are available on the website.

Java Media Framework API (JMF)
http://java.sun.com/products/java-media/jmf/index.jsp

Using Java for media and vision. Says the website: "The Java Media Framework API (JMF) enables audio, video, and other time-based media to

FIGURE 1. This page at **geocities.com/vjkemp/gbcam.htm** is one of several on the Internet that explain how to create a low-cost vision system.

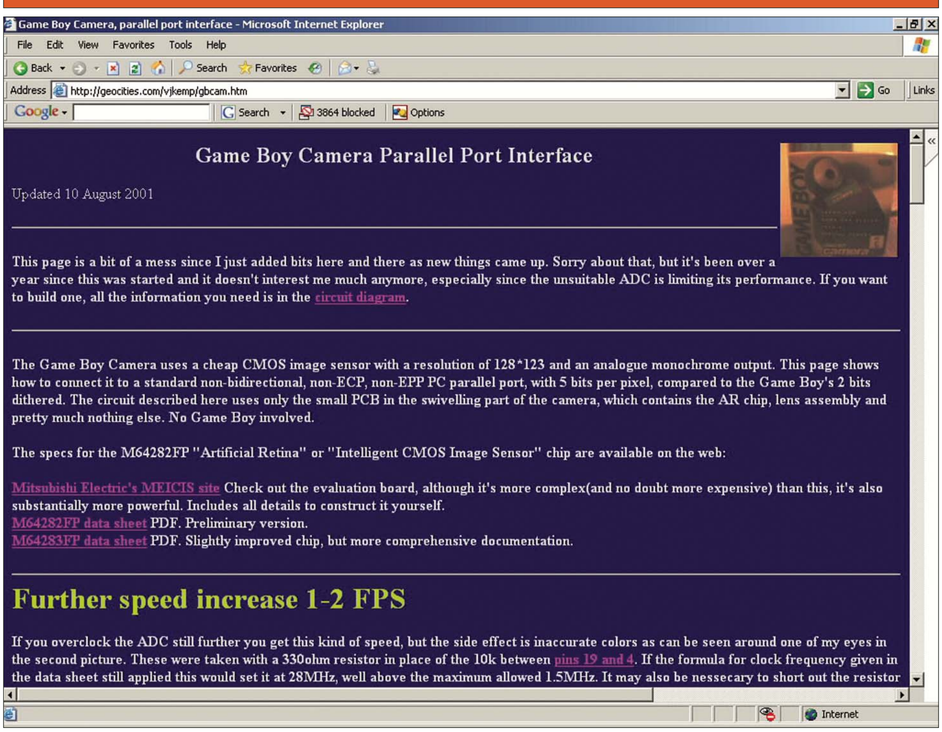


FIGURE 2. OmniVision is one of the world's largest manufacturers of sensors for miniature cameras. Check out their website for datasheets.

FIGURE 3. Cameras galore at PolarisUSA Video — one of the many web-based retailers of security and miniature cameras.

be added to Java applications and applets. This optional package, which can capture, play back, stream, and transcode multiple media formats extends the multimedia capabilities on the J2SE platform and gives multimedia developers a powerful toolkit to develop scalable, cross-platform technology." The JMF is capable of capturing, processing, and displaying data from most any analog camera. Can be used with any computer that supports Java, including Linux and the PC.

Machine Vision Online
www.machinevisiononline.org

Machine Vision Online is the portal for all things machine vision.

Matco Inc.
www.matco.com

MATCO sells CCD security cameras and CCTV surveillance products.

Micro Video Products
www.microvideo.ca

Micro Video sells miniature black-and-white and color video cameras (with integrated lens, though many are removable and interchangeable), bullet cameras, and wireless video systems. All cameras are NTSC (for color) or EIA (for black-and-white) compatible.

OmniVision Technologies, Inc.
www.ovt.com

Single-chip CMOS black-and-white and color imagers — "Single-Chip CMOS Image Sensors (Camera-on-a-Chip)." The company designs single-chip image sensors for capturing and converting images for cameras. Their imagers are used in a number of products and designs, including the CMUcam.

Open Source Computer Vision Library
www.intel.com/technology/computing/opencv/index.htm

The website's aim is to "Aid commercial uses of computer vision in human-computer interface, robotics, monitoring, biometrics, and

security by providing a free and open infrastructure where the distributed efforts of the vision community can be consolidated and performance optimized."

Pelikan Industry, Inc.
www.pelikancam.com

Pelikan provides video surveillance systems, including cameras and video transmitters.

Pixel Smart
www.pixelsmart.com

Pixel Smart sells frame grabber boards for the PC. Includes a free development SDK that can be used to program ways to manipulate the captured images.

PolarisUSA Video
www.polarisusa.com

Polaris Industries supplies security cameras and wireless transmitters for video.

Programming Video for Windows
http://ej.bantz.com/video/detail/

Informational page on using a video camera with Windows, and writing programming code to capture frames with the Video for Windows application interface. For machines with Windows, obviously. (*Note: Later versions of Windows incorporate video playback and capture services in the DirectX interfaces.*)

QuickCam
www.quickcam.com

All about the Logitech QuickCam, including drivers. The QuickCam is one of the most popular PC-based cameras available.

SuperCircuits
www.supercircuits.com

SuperCircuits sells wired and wireless video cameras, miniature (including the size of a shirt button) cameras, and video transmitters and receivers.

Theremin Vision
thereminvision.com

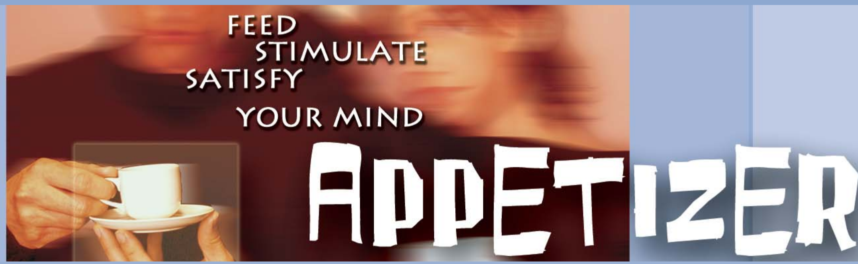
Interesting application of theremin design principles to machine vision.

X10 Wireless Technology, Inc.
www.x10.com

X10 Wireless Technology sells home automation products for wireless remote control. Also line of moderate-quality color and black-and-white security cameras. **SV**

ABOUT THE AUTHOR

*Gordon McComb is the author of the best-selling Robot Builder's Bonanza, Robot Builder's Sourcebook, and Constructing Robot Bases — all from Tab/McGraw-Hill. In addition to writing books, he operates a small manufacturing company dedicated to low-cost amateur robotics, **www.budgetrobotics.com** He can be reached at robots@robotoid.com*



Calling the Future

by Robin Hewitt

I have a clamshell cell phone, and when I flip it open, it reminds me of the communicators in the original *Star Trek* series. For a techno-geek like me, there's always a thrill of excitement in seeing what was once science fiction enter the mainstream of everyday life.

Even some extreme technology from the original *Star Trek* is available now. Do you recall, for example, the omniscient Ship's Computer? Ask it any question, and it would give you the answer. It would even speculate for you! At the time, this seemed preposterous, yet although the interface is different, today's Google and Wikipedia offer a capability very much like that.

Interestingly, the technology for science fiction's humanlike robotic characters has proven more elusive. I don't mean physically imitating the human form. Indeed, in that area, we've seen enormous progress! But it's not primarily the humanoid shape that captures the imagination and inspires us. Rather, it's the more abstractly human qualities — the autonomy we see in *Star Wars*' R2D2, the sense of personality we get from Huey, Louie, and Dewey in *Silent Running*.

For these capabilities to become possible, robots need greater awareness of their environment. Computer vision is one technology that offers potential for increasing robotic awareness. An overwhelming difficulty in developing human-like visual capabilities has always been the enormous processing power required. Visual information is inherently distributed across multiple spatial locations and scales. It's distributed temporally, as well. The number of possible pixel combinations from even a short video vastly exceeds the number of protons in the observable universe. The latter is on the order of 1e88, while cross-comparing all pixel-scale data between 120 640x480 video frames (about four seconds worth) would require $(640x480)^2$ x (119!) = 1.7e207 comparisons!

Over the past two decades, improvements in computer vision have arisen from a combination of faster CPU speeds plus better algorithms. Algorithms continue to improve, but CPU speeds may soon reach hard limits. More importantly, sequential CPU processing is by nature a poor match for what's inherently a parallel problem. Perhaps that's why human visual processing occurs within a markedly parallel, interconnected system.

Although CPUs can be connected to form parallel computing systems, multi-CPU "supercomputers" carry significant power and space overhead. Fortunately, there may be a way out of this difficulty, and the technology to escape it is already here. It's called GPU computing. GPU stands for Graphics

Processing Unit. It's a different type of processor — one that's highly parallel by design.

If you have a computer that will run modern video games, you probably already have a GPU; it's on your video card. While GPUs have impressive computational power that's well suited to visual processing tasks, accessing this power is currently difficult. Programming languages, such as OpenGL, that access GPUs were never intended to support general-purpose computing tasks. They're designed to support rendering — visual display, not vision processing.

But this may soon change. And a major driver for change is ... cell phones! Specifically, it's the competition to combine cell phone and camera technology into a single inexpensive, low-power, palm-sized unit. Already, big video-card manufacturers such as nVidia and ATI are producing GPUs designed specifically for cell phones. Since these cell phones are also cameras, the next logical step is to enable onboard image processing, and where better to do this than on the GPU? Once good software interfaces and programming languages for doing GPU-based image processing become available, we can expect to see impressive advances in intelligent visual processing. Can advanced visual intelligence for robotics be far behind?

It was science-fiction writer Arthur C. Clarke who first proposed the geostationary satellites — satellites placed in a special orbit to keep them positioned over the same spot on Earth — that were the keystone in creating today's global wireless communications web. Today, thanks to this satellite technology, we have not only cell phones, but wireless Internet and GPS, as well.

Robotics is already making good use of GPS technology. But we've barely begun to tap the potential of wireless Internet for advanced robotics. In visual learning, for example, a human typically prepares numerous examples of the object or paradigm to be learned. This process is laborious, and is usually done by several students. For individual robot owners, this would likely be too much work. But what if the information were available online? Even better, what if it were available in a form that the robot could find for itself!

Internet-based data exchange needn't be limited to visual data. Behind Google's search engine are databases of word associations, frequency counts, and relevancy statistics. What if this information were available to robots learning linguistic concepts?

Perhaps someday, parents commuting home on intelligent transport will use that time to call their child's teacher from something that's a combination cell phone, bank card, and portable personal assistant — *with* configurable personality. Afterwards, the assistant might ask if it should tell the home educational robot to search online for options to train itself as an algebra tutor. Meanwhile, the transport vehicle might be updating a robot-wiki of road-hazard statistics by posting summaries from its daily video feed ... **SV**

Then and NOW

MOVIE ROBOTS

by Tom Carroll

All of us who are into experimental robotics have certainly enjoyed watching our share of movie robots on the big screen, and were probably inspired by one or more as a child. Movie robots are "bigger than life" and certainly seem to do more than any robot we've ever seen in real life. As we all know, virtually none of them are actually real, working robots, but are what is known in the movie business as "action props," or even computer-generated images for the latest films.

It wouldn't do any good to jump right into the more modern action prop robots without looking back a few more years to see just how early writer's creations made their first appearances on stage and film. The very first film robots were from the mythical Jewish stories about the Golem. There are many versions of the legend of the Golem, but the most widely known version takes place in Prague during the 16th century.

Rabbi Loew was the leader of the Prague Jewish community. The Jewish people had been wronged, so Loew animated a man made from clay to extract revenge on supposed wrongdoings. What the Golem did and how they reacted varies widely with different movies and tales. Three films were made about the Golem from 1914 to 1920 (Figure 1). In this period, many very advanced clockwork mechanisms had been in existence for centuries, but the makings of a mechanical human was so mysterious that the Golem's insides were left to the audience's imaginations. They were depicted as mindless monsters with clay hair.

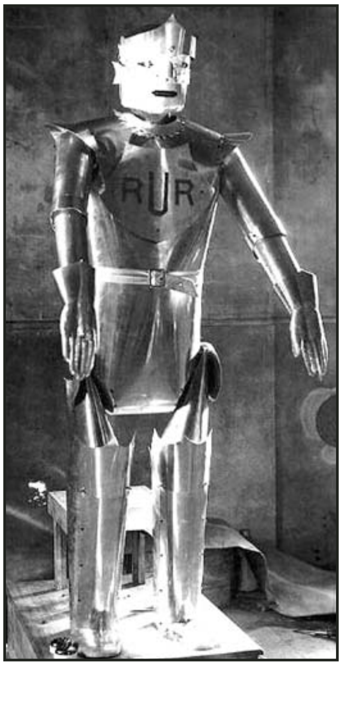
The true turning point in the development of mechanical humans in literature was the play *RUR* (Rossum's Universal Robots) by Czech playwright, Karel Capek in 1919. First performed in Prague in 1920, the "robots" in the play looked pretty much like people. Helena Glory — the robotess made in the likeness of a professor's daughter — is one of the characters, and is shown in Figure 2. Maybe if you walked stiffly and talked a bit strange, a normal-looking actress could pass for a robot in those days.

Figure 3 shows an actor in a metal suit from one of the later US stage versions of *RUR*. This robot was still considered to have biological "insides" with a metal skin. The most unique thing to come from this play is our word — robot. Robota meant drudgery or servitude and robotnik meant a peasant or serf. The *RUR* robots ended up taking over the humans, and as a final insult, asked the last human just how could they reproduce. You

Figure 1. Golem.

Figure 2. Helena from *RUR*.

Figure 3. Later version from *RUR*.



should read this play sometime.

In 1927, Fritz Lang produced a silent movie in Germany that is considered, by most, as an all-time classic. *Metropolis* featured a robot — Maria — who was created by Rotwang, an assistant to Joh Frederson, the Master of Metropolis. She was made in the likeness of Maria, one of the worker's daughters, just like in *RUR* (Figure 4).

Lang was well ahead of his time showing many yet-to-be inventions in his film. Filmmaker George Lucas is said to have created his *Star Wars* robot — C3PO — after seeing Maria. The skillfully crafted body suit is distinctly mechanical in nature. Except for the obvious feminine features and expressionless face, she certainly bears a distinct resemblance to C3PO.

Movie robots have now changed from semi-biologic creatures created by mystical means to sophisticated machines created by scientists, even though they could be considered "mad scientists." Gort in *The Day The Earth Stood Still* — the 1951 B&W classic — is also a noted movie robot. This robot was actually a 7'7" actor — Lock Martin — in a rubber "robot" suit. The filming was a miserable experience for him with many hot hours covered in rubber. The line "Gort. Klaatu Barada Nikto" is the famous line that is to be given to Gort by Helen (yes, another Helen) the female love interest to stop the robot from destroying the Earth. Klaatu, the alien gave these words to her as he lay dying (see Figure 5).

Movie robots finally came of age in 1956 with the release of *Forbidden Planet*. This 1950's science fiction box office success containing many technological feats such as faster than light travel, an alien race (now extinct), and even ray guns that didn't shoot sparks and smoke. People may have forgotten Anne Francis (see Figure 6) — the young love interest actress in the movie and the ship's commander played by Leslie Nielsen, whom we now remember most as the bumbling police detective, Frank Drebin in the *Police Squad* series. It is Robby the Robot of *Forbidden Planet* that all of us remember the most (see Figure 7).

Robby was designed by Robert Kinoshita and built in mid-1955 by the MGM prop department at a cost of $125,000. Robby has become one of the most popular movie robots in the history of movies, and is as recognizable as *Star Wars*' R2-D2 and C-3PO. Robby was conceived with a semblance of Asimov's Three Laws to protect human beings. Robby was about 7' 6" tall and weighed about 300 lbs. MGM special-effects technician, Glen Robinson, operated him and actor and announcer, Marvin Miller, provided his voice. Actor and stunt man, Frankie Darro inside the robot's shell accomplished most internal motions. His head did contain what looked like a bunch of twirling tops and relays that sounded like a mechanical adding machine — a first for action props. Film director, Bill Malone, now owns Robby.

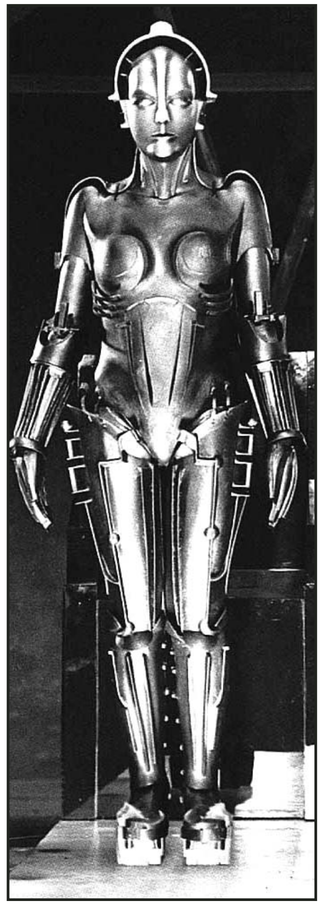


Figure 4. Maria from *Metropolis*.

Figure 5. Gort and Klaatu.

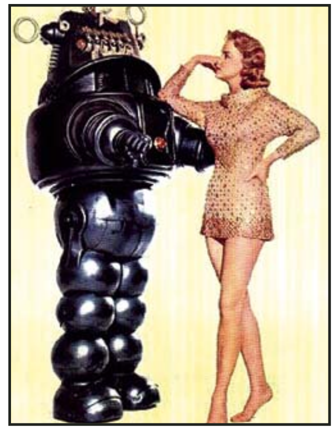


Figure 6. Robby and Anne Francis from *Forbidden Planet*.

Jumping 30 years forward to *Revenge of the Nerds* produced in 1984, action props now used robotic mechanisms instead of robot "suits." Living in the LA area, I was fortunate to be able to design and produce all four robots used in this film in my garage in Long Beach. Working with Joe Unsinn, special-effects supervisor, and my friend, Dail DeVilleneuve of Vantec, under deadline we completed two fully-functional and two semi-functional robots in about three weeks. These machines had functional arms (shoulder and elbow joint) and used standard 75 MHz radio control systems. I had also earlier built a robot from a blue plastic drum that was used in the third movie of this series.

Short Circuit, produced in 1986, was probably the finest example of a truly functional teleoperated robot that was used in a movie. These machines were handcrafted by Eric Allard, robotics special-effects supervisor, with Gary Foster and Dennis Jones. It took the team up to five weeks to make each robot, especially the more complex "Number 5s." Fifteen robots were

Figure 7. Robby the Robot.

made, including extra hands and arms that were held by special-effects people off camera for close-up shots of Number 5 handling something.

I have in no way covered everyone's favorite robot from TV and the movies. The robot from *Lost in Space*, the two robots from *Buck Rogers and the Twenty-fifth Century*, Huey, Dewey and Louie from 1972's low budget *Silent Running*, and many others are cult favorites. With the advent of powerful computer workstations, it became easier and cheaper to create images of robots rather than complex mechanisms. The 2004 Will Smith film — *I Robot* — is a good example.

There are just too many robots running around to create a physical prop to represent each robot. The 1,000 robots in a large room waiting to be delivered is a great example. I furnished about a dozen robots and robot parts for an "antique robot store front" (remember, the movie takes place in 2035) for the filming in nearby Vancouver area, and all the shots of the store front ended up on the cutting room floor. Computer imaging has taken over the prop shop's work as far as robots go, a long ways from the stop motion action props of 80 years ago. **SV**

ABOUT THE AUTHOR

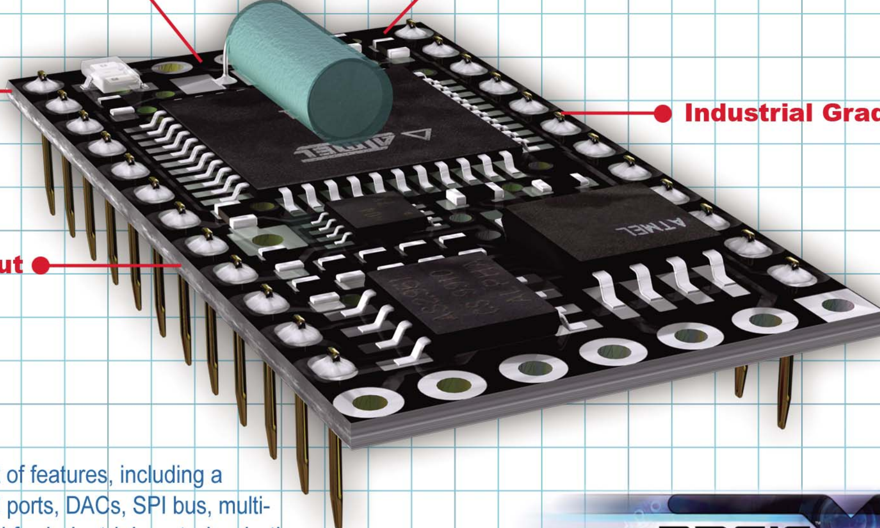
Retired from Rockwell and NASA projects, Tom Carroll is a space robotics engineer. He's authored numerous articles on combat robots, lives on an island in the Pacific Northwest with his wife, Sue, and enjoys robotics, kayaking, and hiking. He can be reached at: TWCarroll@aol.com

Advertiser Index

Announcing the BasicX - BX24-P

The *fastest* Basic programmable microcontroller...

Just Got BETTER for the same price $49.95



FEATURES: The BX24-P has a vast set of features, including a full IEEE floating point math library, COM ports, DACs, SPI bus, multi-tasking and networking, making it ideal for industrial control, robotics, automated testing equipment and home automation. BX24-P is able to monitor and control all your switches, timers, motors, sensors, relays, and more. Measuring just 1.25" X .75", OEMs can easily see the value of this little powerhouse.

WWW.BASICX.COM

CALL 520.544.4567 FOR QUANTITY PRICING

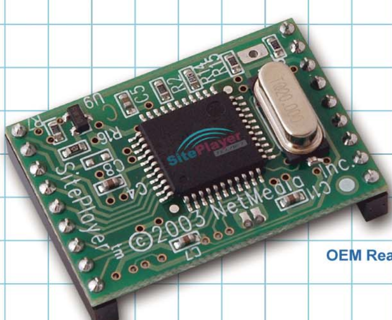
Announcing SitePlayer Telnet™ Serial to Ethernet Data Converter

Enable your serial devices to communicate across Ethernet

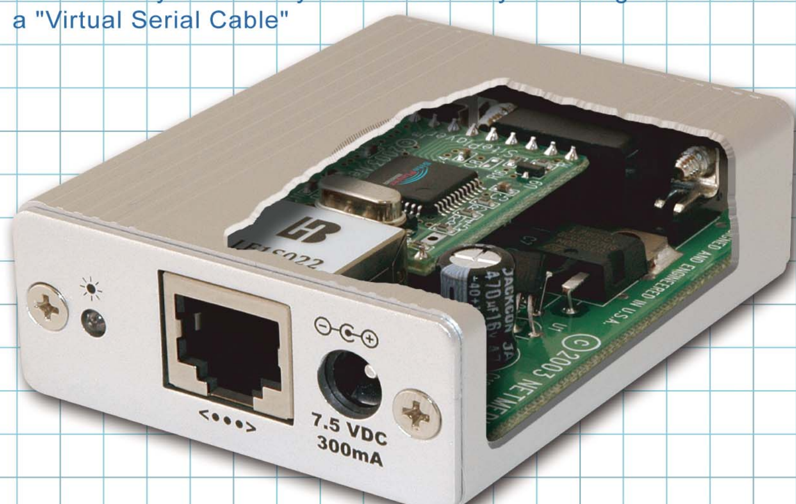
The SitePlayer Telnet module gives OEM's and product designers the ability to quickly and cost-effectively bring their products to market

Two SitePlayer Telnet systems can easily be configured to create a "Virtual Serial Cable"

SPT1 – SitePlayer Telnet™ OEM module
ONLY $29.95



OEM Ready



Features - 10BaseT, ARP, UDP, TCP/IP, DHCP, Link Local, Bonjour, ICMP Ping, HTTP, Telnet, Daytime protocol, Discard protocol, 4 IP addresses, Serial AT command mode for external processor control of Ethernet sessions.

SPTS – SitePlayer Telnet™ System, now available directly from NetMedia at an **Introductory price of $79.95**

NetMedia Inc., 10940 N Stallard Pl, Tucson Arizona 85737 tel.520.544.4567x101 fax.520.544.0800 www.siteplayer.com